

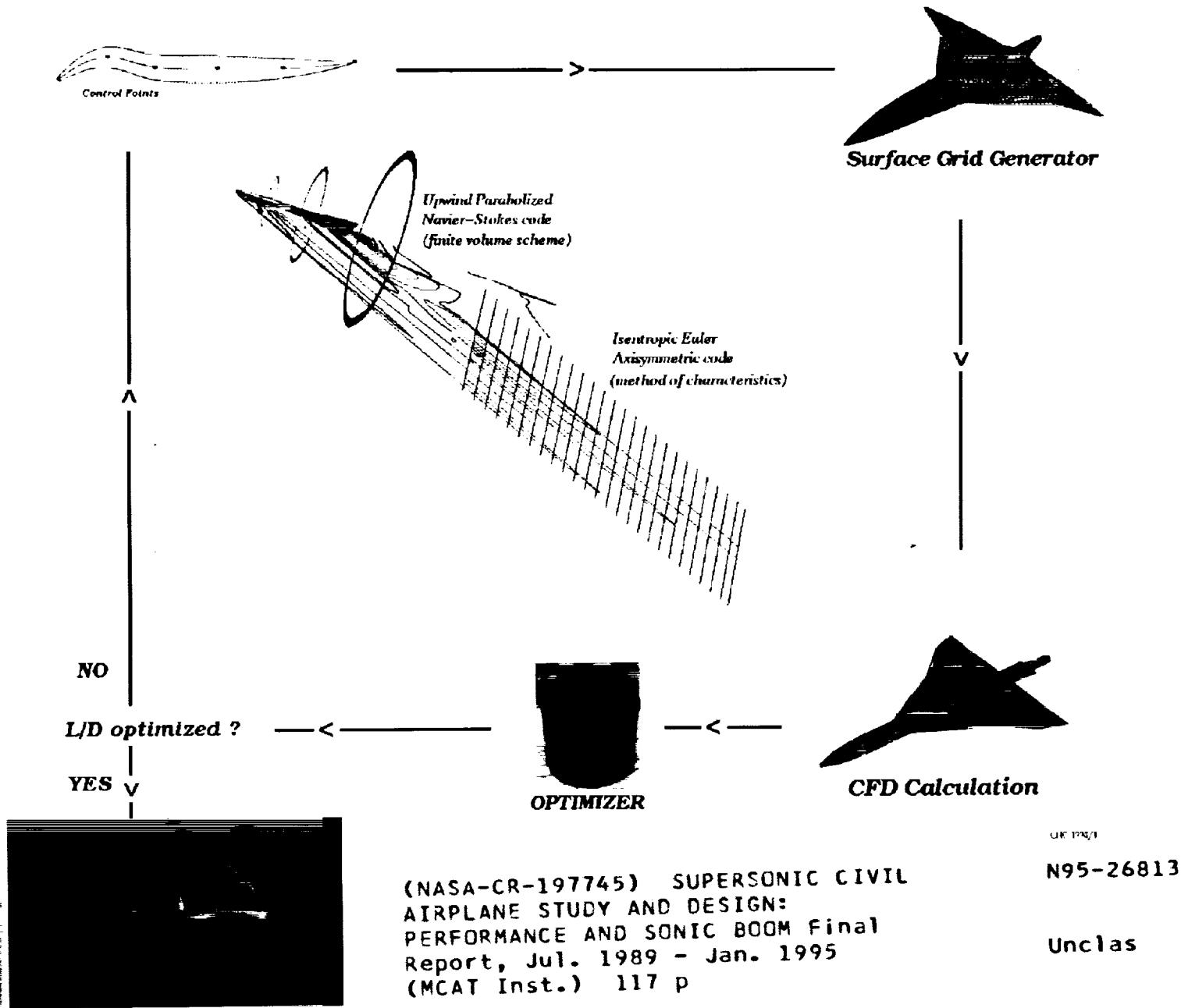
**MCAT Institute  
Final Report  
95-7**

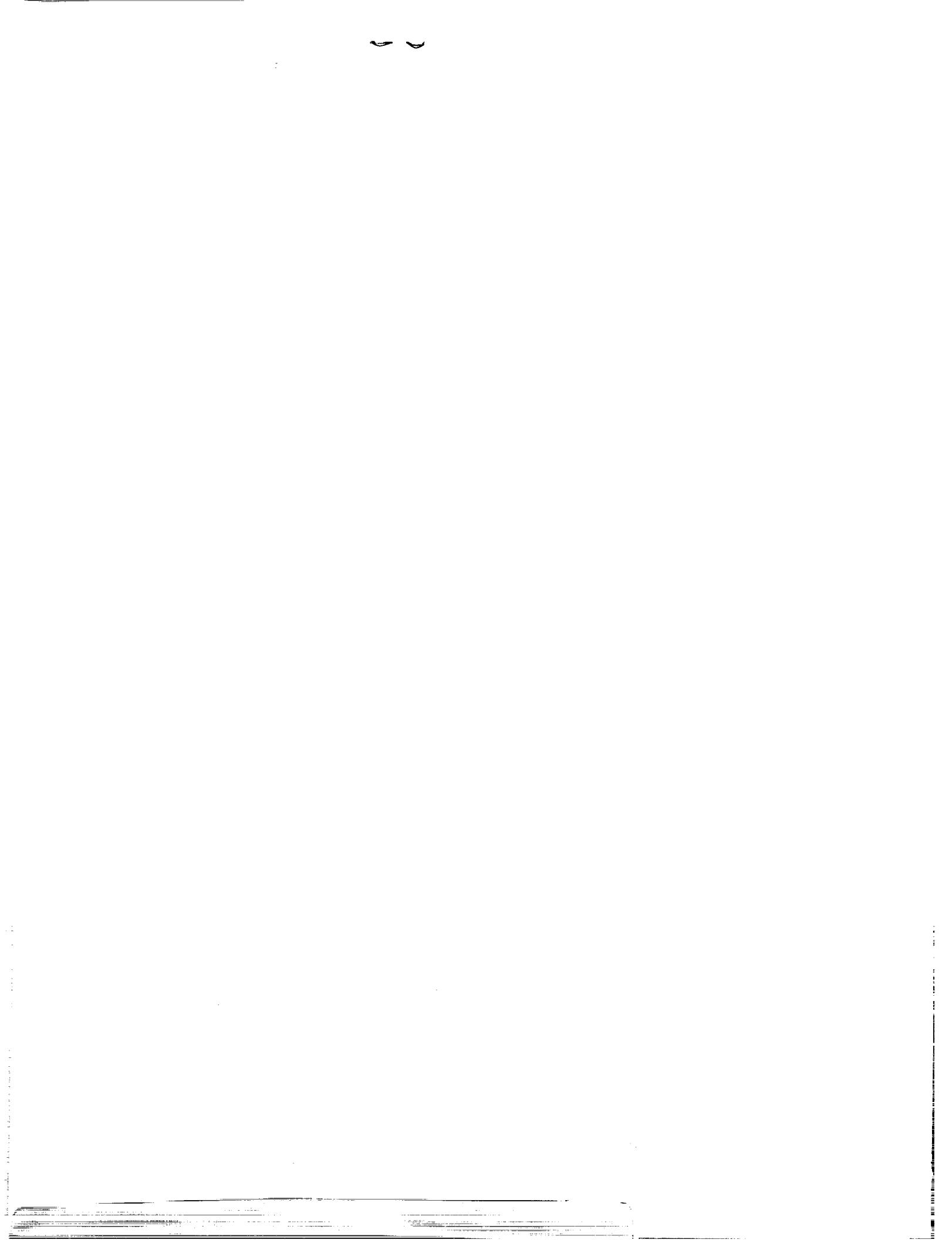
**MCAT Institute  
3933 Blue Gum Drive  
San Jose, CA 95127**

# Supersonic Civil Airplane Study and Design: Performance and Sonic Boom

**Samson Cheung**

**January 1995  
NCC2-617**





# Supersonic Civil Airplane Study and Design: Performance and Sonic Boom

Samson Cheung

This final report summarizes the work performed from July 1989 to Jan. 1995. The work is supported by NASA Co-operative Agreement NCC2-617. This report consists of four parts. The first part is the introduction of the research effort. The second part describes the work and results from July 1989 to June 1993. The third part describes the work and results from July 1993 to January 1995. A summary is given at the end of this report.

## 1 INTRODUCTION

The present supersonic civil airplane, the Concorde, is a technological break-through in aviation history. However, it is an economical disaster for two main reasons. The first is her low aerodynamic performance, that allows only 100 passengers to be carried for a short-range flight with expansive airfare. Another reason is that the shock waves, generated at supersonic cruise, coalesce and form a classical N-wave on the ground, forming a double bang noise termed sonic boom, which is environmentally unacceptable. To enhance the U.S. market share in supersonic civil transport, an airframer's market risk for a low-boom airplane has to be reduced.

Since aircraft configuration plays an important role on aerodynamic performance and sonic boom shape, the configuration of the next generation supersonic civil transport has to be tailored to meet high aerodynamic performance and low sonic boom requirements. Computational fluid dynamics (CFD) can be used to design airplanes to meet these dual objectives. The work and results in this report are used to support NASA's High Speed Research Program (HSRP).

In this five years of study and research, CFD tools and techniques have been developed for general usages of sonic boom propagation study and aerodynamic design. In the beginning of the 90's, sonic boom extrapolation technique was still relied on the linear theory developed in the 60's for the nonlinear techniques were computationally expensive. A fast and accurate sonic boom extrapolation methodology (Section 3.2), solving the Euler equations for axisymmetric flow, has brought the sonic boom extrapolation technique up to the 90's standard.

Parallel to the research effort on sonic boom extrapolation, CFD flow solvers have been coupled with a numeric optimization tool to form a design package for aircraft configura-



tion. This CFD optimization package has been applied to configuration design on a low-boom concept (Section 2.3) and an Oblique All-Wing concept (Section 2.4) prior to the wind-tunnel models are built and tested at Ames. The tunnel test results have validated the CFD technique and design tools.

Moving to the world of parallel computing, the aerospace industry needs a numeric optimization tool suitable for parallel computers. A nonlinear unconstrained optimizer for Parallel Virtual Machine has been developed for aerodynamic design and study. Study in Section 3.3 demonstrates the capability of this optimizer on aerodynamic design.

## 2 PREVIOUS WORK/RESULTS

The work and results described in this section was begun in July 1989. The first project was to use CFD tools and existing linear theory to predict waveform signatures at some distances from flight vehicles. The aim of this study was to demonstrate and develop the technique of sonic boom prediction by CFD. The next step was to apply this developed technique to low-boom configurations.

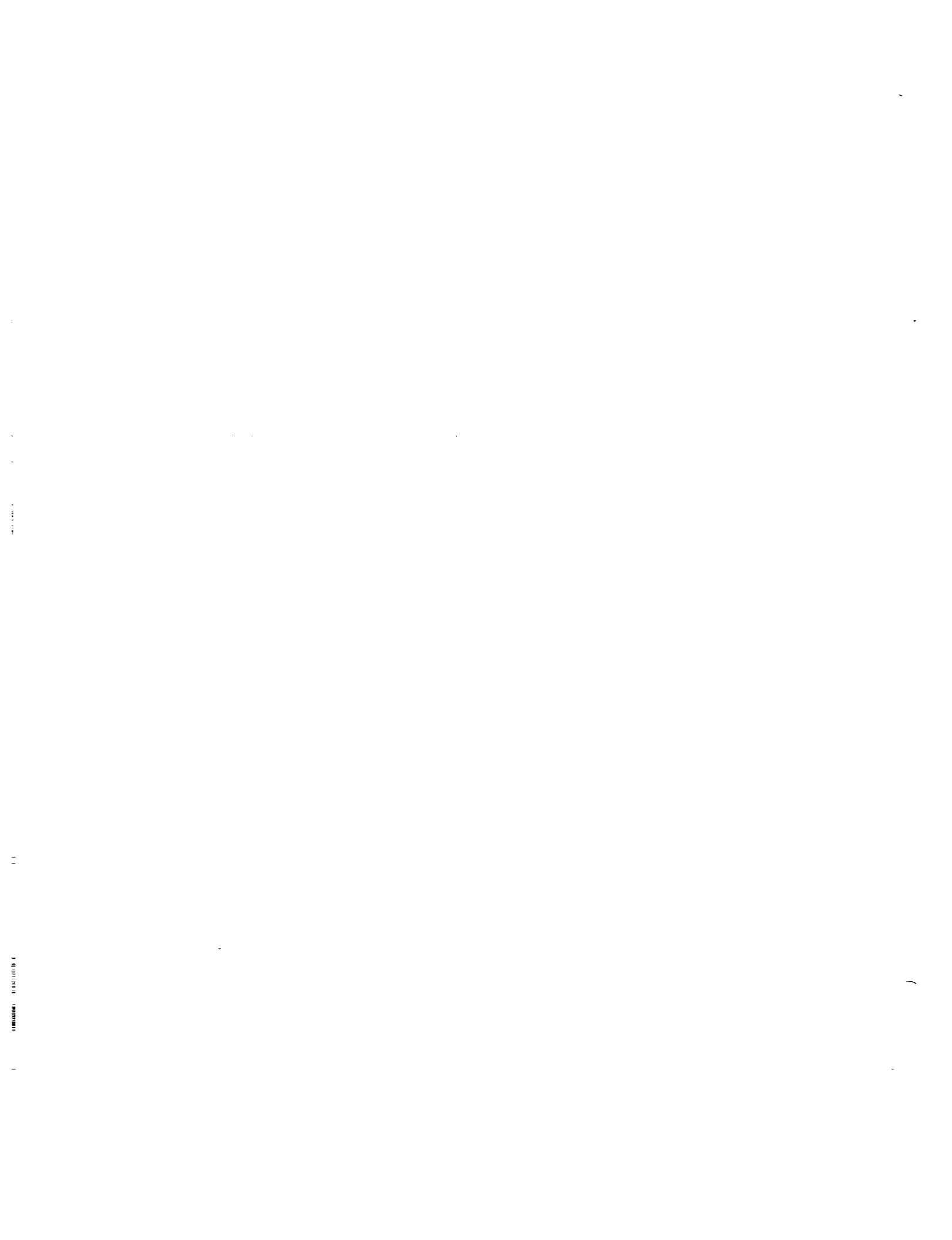
The second project, which was the continuation of the first one, was to develop a CFD optimization package for design process on meeting the dual objectives of high aerodynamic performance and low sonic boom loudness. This optimization package was applied to three different High Speed Civil Transport (HSCT) baseline configurations and a generic body of revolution.

A wind-tunnel model (Ames Model 3) was built based on one of the modified HSCT baseline configuration. This model was tested in June 1993. The test results were used to validate the design method. Publication of the result was limited due to the sensitive nature of the project.

A counterpart of the conventional HSCT concept was the Oblique All-Wing (OAW) concept. CFD computational supports, as well as optimization calculations, were provided to the OAW design team consisting personnels from NASA Ames Research Center, industry, and university. The aim of the project was to design a realistic configuration for wind-tunnel test. The model was built and tested at Ames in June 1994.

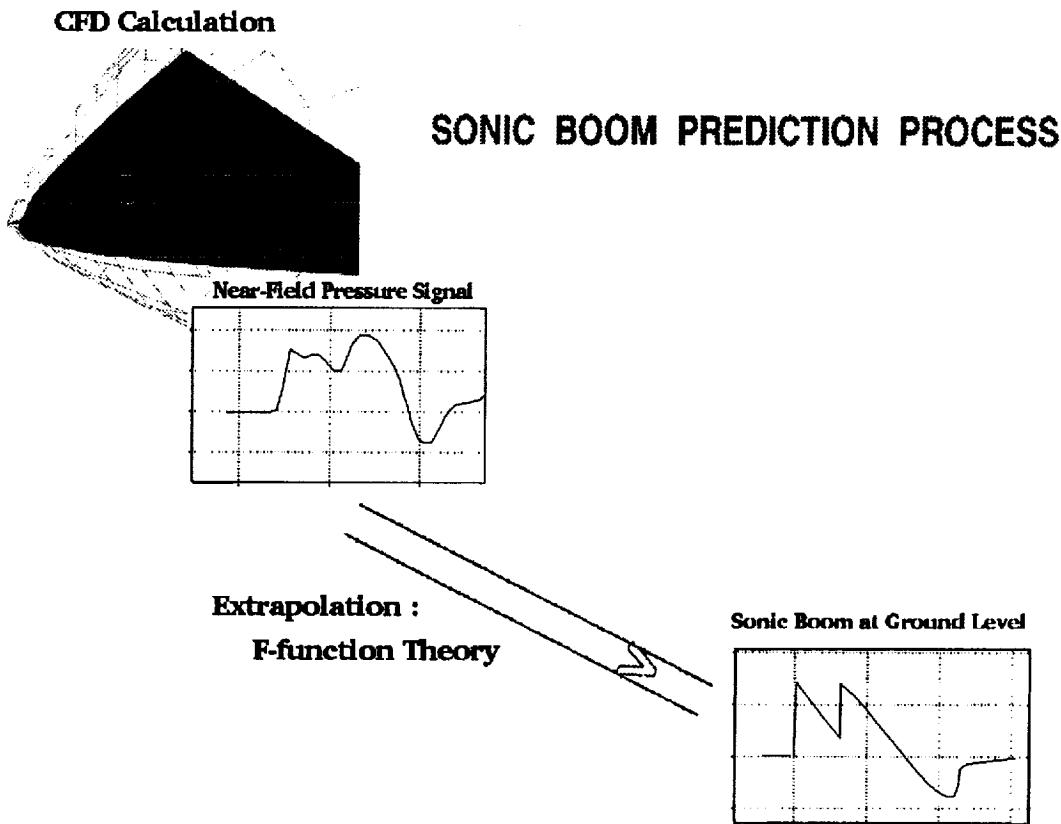
### 2.1 Sonic Boom Prediction Technique

In the early stage of sonic boom prediction activity, two major things were involved. The very first thing was to identify the capability of CFD in sonic boom prediction. The second thing was to apply these CFD tools to predict sonic boom signals of varies configurations after necessary code modification, grid refinement study, and comparison with supersonic linear theory.



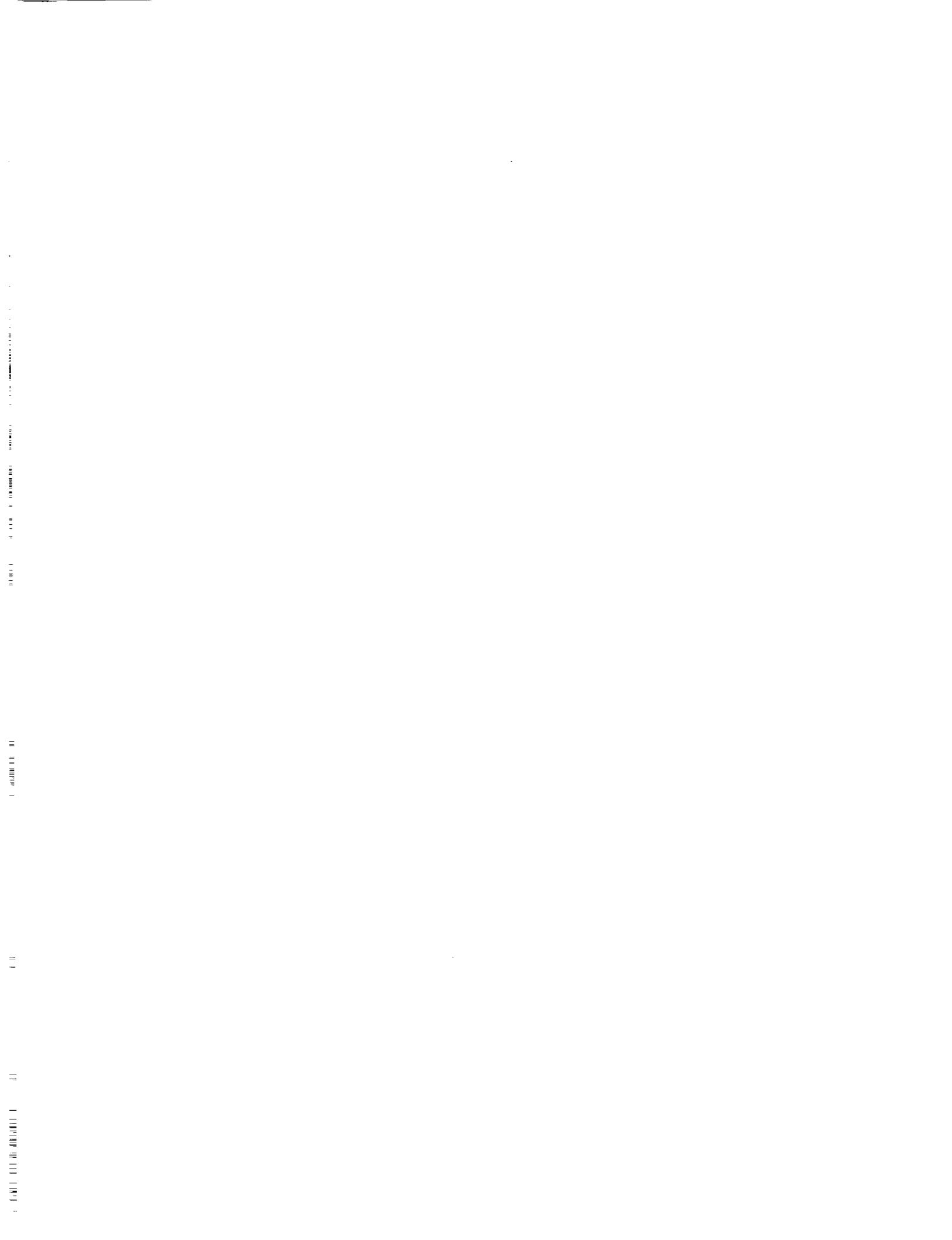
### 2.1.1 Method Validation

A three-dimensional parabolized Navier-Stokes code, UPS3D,<sup>1</sup> developed at Ames was used as the flow-solver. It is a space-marching code with finite-volume approach. The near field solution of a simple wing/body configuration was calculated by UPS3D, and the overpressure signal at some desired distances were obtained either by the axisymmetric option of UPS3D or a quasi-linear extrapolation code, based on Whitham's F-function theory<sup>2</sup>. Later I realized that using Lighthill integral<sup>3</sup> to calculate the F-function for non-axisymmetric aircraft was more accurate, I wrote a Fortran code, LHF, for sonic boom prediction based on Lighthill integral. This code is available from Ames Software Library. A copy of LHF is attached in Appendix A. The figure below is a brief summary of the sonic boom extrapolation process.



A series of studies on grid refinement, including solution adaptive grid, and on sensitivity of initial distance of extrapolation were conducted. It was found that viscous calculation was unnecessary for sonic boom prediction. However, the grid must be sufficiently fine in the regions of shock and expansion waves. In order to capture all the nonlinear effects in a three-dimensional flow, the near-field overpressure should be captured at about one span length below the flight track before extrapolating to the far field. The detail results were published in AIAA Journal of Aircraft<sup>4</sup> and NASA Technical note<sup>5</sup>.

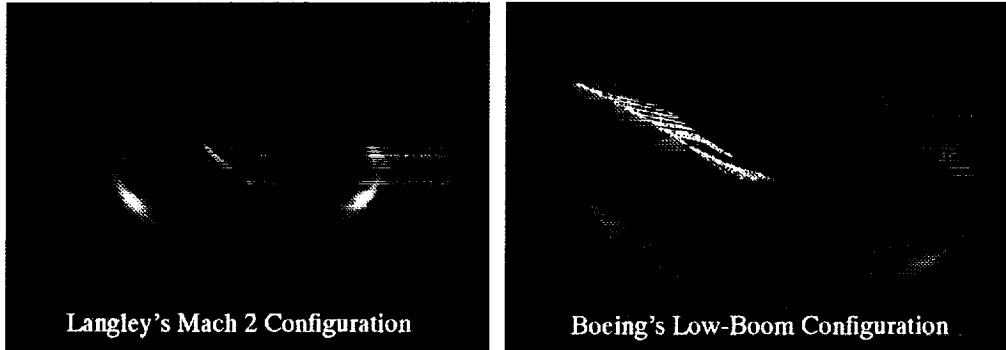
In summary, the tools for sonic boom prediction had been identified and validated in the above study. The combination of CFD and Whitham's method gave a relatively efficient tool for sonic boom prediction. Nevertheless, the CFD codes were still computationally expensive for design optimization runs.



### 2.1.2 Boom Prediction for Low-Boom Configurations

With the experience on grid refinement study and the extrapolation procedure, the prediction tools were being used to predict the sonic boom of two low-boom configurations designed by Boeing aircraft company and Langley research center.

Each of the two configurations consisted of two separated parts, namely, the wing and the fuselage. The wing was defined by data in spanwise cuts, whereas the fuselage was defined by data in streamwise cuts. In order to create a single wing-fuselage surface grid for UPS3D code, a grid generator (SAMGRID) was written to define the wing in streamwise cuts and aggregated the wing to the fuselage. Computation results of the two configurations are shown below.

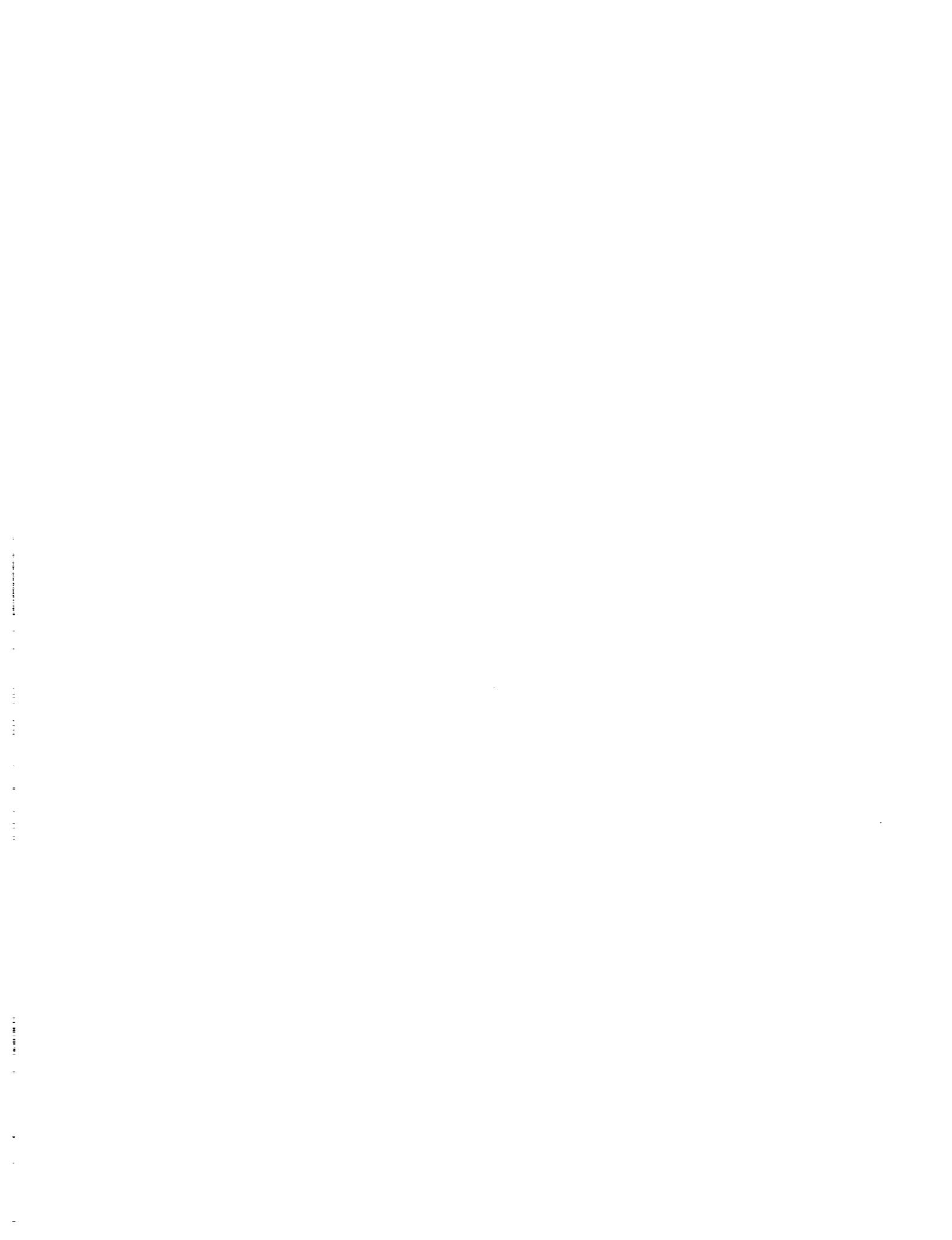


The sonic boom signals calculated from the CFD prediction tools were compared to the wind-tunnel data of the Langley's configuration. The computational results of the Boeing's configuration was used to validate the linear design method used by Boeing.

## 2.2 Supersonic Airplane Design

The need for simultaneous sonic boom and aerodynamic optimization was highlighted when it became clear that designed to a strict sonic boom constraint suffered an unacceptable performance penalty. Therefore, low-boom design studies must carefully balance the trade-off between sonic boom loudness and aerodynamic performance. A CFD optimization package was developed to demonstrate the methodology for the optimization of supersonic airplane designs to meet the dual objectives of low sonic boom and high aerodynamic performance.

In this project, an optimizer with linear and nonlinear constraints was first identified, and then an efficient CFD flow solver was chosen. This CFD code had to be sufficiently fast because more than 90% of the computational time were used in CFD calculations. Before this optimization was used to design low-boom wind-tunnel model (Section 2.3), it was tested and exercised by improving aerodynamic performance of a low-boom wing/body configuration and a body of revolution.



### 2.2.1 CFD Optimization Package

Several computational tools interconnect in the optimization procedure are listed below:

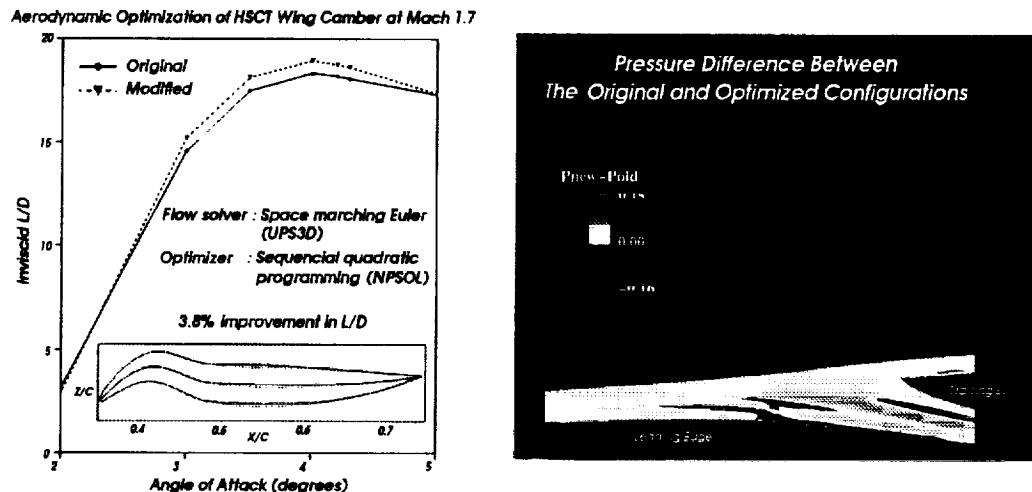
- UPS3D: 3-D parabolized Navier-Stokes code; inviscid calculation only (Ref. 1)
- NPSOL: numerical optimization code<sup>6</sup>; a sequential quadratic programming algorithm in which the search direction is the solution of a quadratic programming subproblem
- HYPGEN: hyperbolic grid generator<sup>7</sup>; a sufficiently fast and robust to operate within an automated optimization environment.
- LHF: sonic boom extrapolation code (Appendix A); a routine based on Whitham's F-function and the equal-area rule<sup>8</sup>
- SAMGRID: wing/body surface grid generator (Appendix B); a sufficiently fast and robust to operate within an automated optimization environment
- DB: sonic boom loudness calculation; a code gives perceived loudness (PLdB) of the sonic boom can be determined by Stevens' Mark VII method<sup>9</sup> which involves Fast Fourier Transform on the energy spectrum of the sonic boom

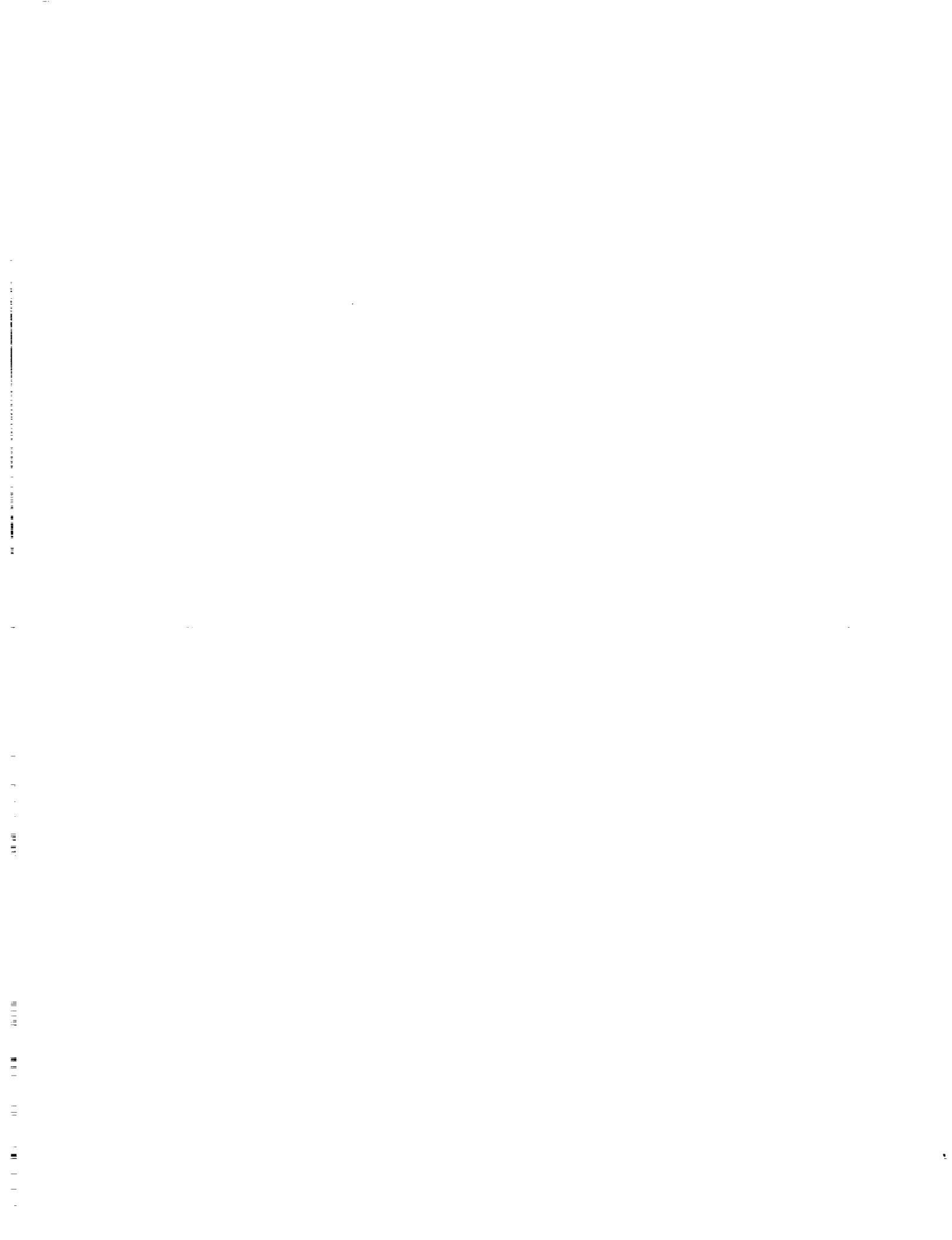
This CFD optimization package is robust and efficient on Cray-YMP. The application of this package will be described in the following sections.

### 2.2.2 Aerodynamic and Sonic Boom Optimization

The optimization design package was exercised using a recently-developed low-boom wing-body configuration, Boeing 1080-991 (also called Haglund model), designed by George Haglund. This optimization technique was applied separately to the two objectives of high aerodynamic performance and low sonic-boom loudness.

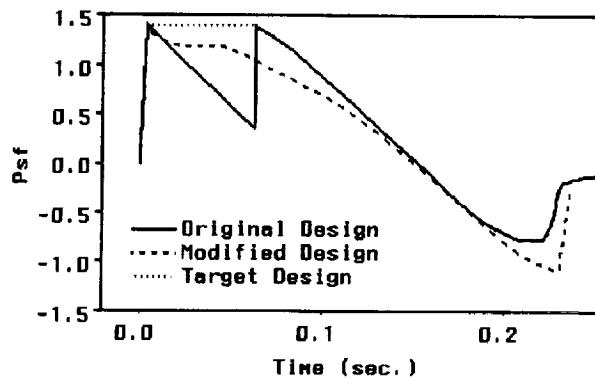
For aerodynamic enhancement, control points are set on the cambers of the wing, with the thickness kept fixed. The left figure below shows the differences on a inboard airfoil section of the original and the modified. The polar plot shows the improvement of L/D of the modified configuration over the original by 3.8%. The right figure below shows that the modified wing had less wave drag than the original one at the leading edge. This means





that the leading thrust is improved by the optimization process. The whole process takes about 4 CPU hours on Cray-YMP.

For sonic boom improvement, F-function was employed as an entity to define the equivalent area distribution and sonic boom shape. The original Haglund model was supposed to give a flat-top pressure waveform at the ground. However, calculations showed that the waveform had an intermediate shock followed right after the bow shock; whereas the flat-top waveform would have no intermediate shock. The design code redistributed the equivalent area of the fuselage (without changing the wings), and re-captured the flat-top characteristic of the pressure waveform. The figure below compares the sonic boom signatures among the original, optimized, and target flat-top. Due to the sensitive nature of the con-



figuration, the change of the configuration will not be shown here. The details of this optimization methodology and results were considered as sensitive materials and were presented in the 2nd Annual Sonic Boom Workshop.<sup>10</sup>

### 2.2.3 Drag Minimization on Haack-Adams Body

The purpose of this study was threefold:

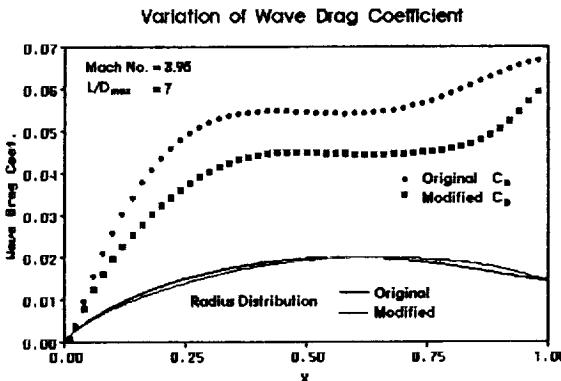
- to search for a design method to minimize the drag of a supersonic projectile
- to demonstrate the capability of the CFD optimization package described above
- to search for computational grid density effect on optimization performance

The baseline configuration chosen for this study was called Haack-Adams body<sup>11</sup>, a body of revolution with a pointed nose and a base of finite area. This body was thought to be the minimum-drag body under the slender body theory. Wind-tunnel data were available for CFD validation. The method of optimization made use of the Fourier Sine expansion, which had three main advantages over the traditional techniques based on shape functions and control points:

- The volume of the body was fixed without putting external constraints. External constraints cost more computational time. For some cases, fixed volume is not feasible.
- Global minimum was search.
- Number of design variables was substantially reduced.



The figure below summarizes the result of this study. The nose of the body was trimmed to

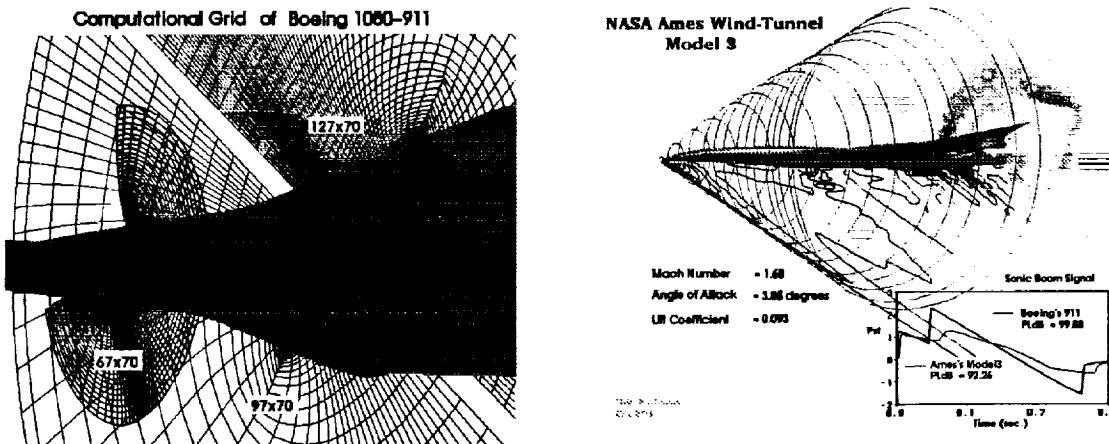


ORIGINAL PAGE  
COLOR PHOTOGRAPH

reduce the wave drag. Since the total volume was constrained, volume was added near the end of body. Total wave drag reduction was by 6%. The results were presented in a AIAA meeting<sup>12</sup> and published in Journal of Aircraft Vol. 32, No. 1, Jan/Feb. 1995.

### 2.3 Low-Boom Wind-Tunnel Configuration (Ames Model 3)

Efforts were made to design a new wing/body/nacelle configuration, which had a lower sonic boom relative to the baseline, 1080-911 from Boeing Company, of low boom HSCT concept. The CFD optimization package described in Section 2.2.1 were employed to modify this baseline configuration. The result of the optimization was used to build a wind-tunnel model, Ames Model 3, tested at Ames 9'x7' wind tunnel in June 1993. Due to the sensitive nature of the configuration, no planform shapes will be shown here. However, the left and right figures below show the computational grid and the optimization result, respectively. The plot at the lower right-hand corner of the right figure shows the

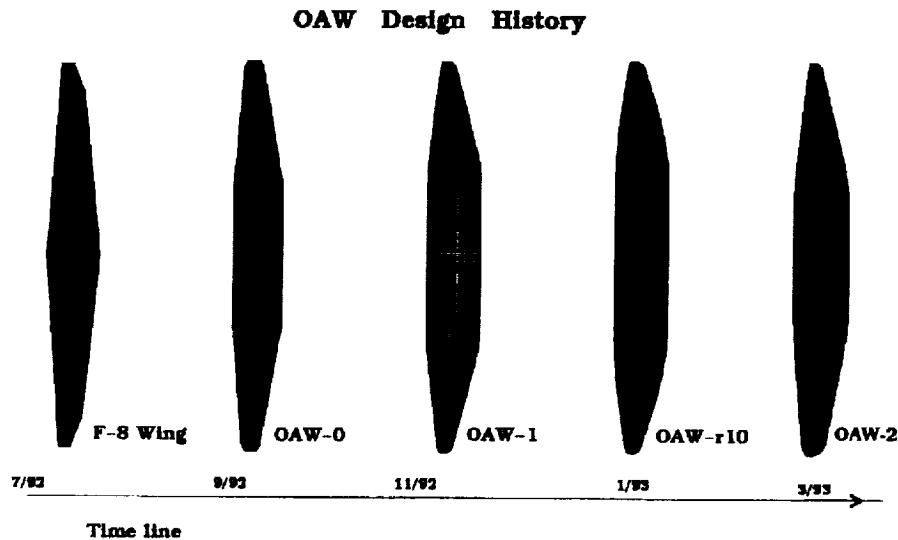


sonic booms of the baseline and Model 3 respectively. The baseline configuration has a loudness level about 100 PLdB; whereas Model 3 has about 92 PLdB. The results of this research were presented in the 3rd Annual Sonic Boom Workshop.<sup>13</sup>

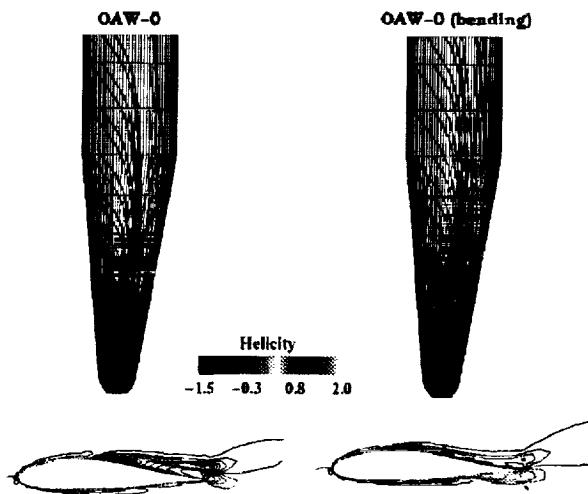


## 2.4 Oblique-All Wing (OAW) Computation and Design

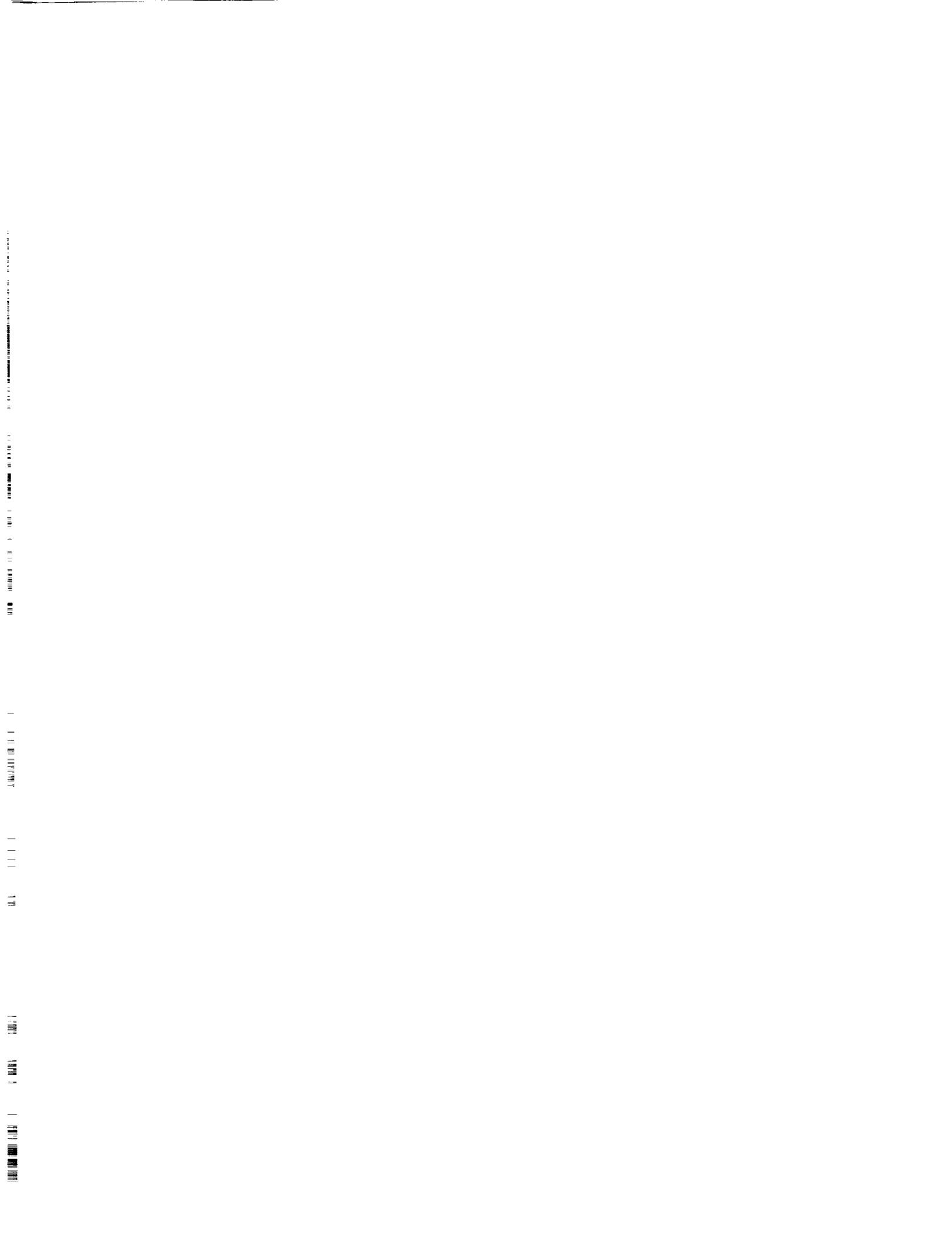
Oblique flying-wing<sup>14</sup> is an alternative supersonic aircraft concept. Ames, Boeing, Douglas, and Stanford University joined and formed a design team in 1992 to investigate the feasibility of OAW for commercial use. The study included aerodynamic performance, stability, structure, landing gear, airplane exits, and airport regulations. The design team decided to build a wind-tunnel model for wind-tunnel testing in June 1994. My job was to provide Navier-Stokes CFD supports and, if possible, optimization results. The figure below shows some of the wings that were analyzed since the beginning of this study.



The flow solver being used was Overflow code, a 3-D Navier-Stokes code using the diagonal with ARC3D algorithm<sup>15</sup>. One of the most challenging works of this project was to reduce the separation on the left wing (trailing wing). The separation on the upper surface of the wing and the corresponding vortices are shown in the left side of the figure below. It



was found that bending of the wing could abate the separation, as well as improve the lift-to-drag ratio. The right side of the figure shows a weaker separation pattern on the ended



wing. Due to the sensitive nature of this study, the results can only be presented in the weekly group meetings at Ames and a controlled distributed NASA Contractor Report.

### 3 CURRENT WORK/RESULTS

Currently, research effort was concentrated on one theme that is sharpening the tools for HSCT design. Three research topics are focused: near-field CFD calculation and sonic boom softening of Boeing Reference-H, improvement of sonic boom extrapolation, and aerodynamic design on parallel computer.

In order to study and design a real complex aircraft, a relatively fast CFD technique has to be developed for optimization environment. Coupling a fast space-marching code and a time iterative code with overset grid concept can take the advantage of marching code at the fuselage/wing region and solve the complex flow field near the wing/nacelle region at the same time.

A very efficient wave propagation code for mid-field sonic boom prediction has been developed based on the method of characteristics. This code solves the Euler equations for 1.2 minutes on Cray-YMP; whereas, the axisymmetric CFD method described in Section 2.1.1 takes 40 minutes on the same computer.

Number crunching problems, like CFD calculations, on parallel machines can be efficiently done in today's computing environment. This may lead to the future of aerodynamic research and design. In order to exercise HSCT design on parallel computers, a nonlinear optimization routine has been developed for a network based parallel computer system in which a cluster of engineering workstations serves as a virtual parallel machine.

#### 3.1 Sonic Boom and Performance Study of Reference-H

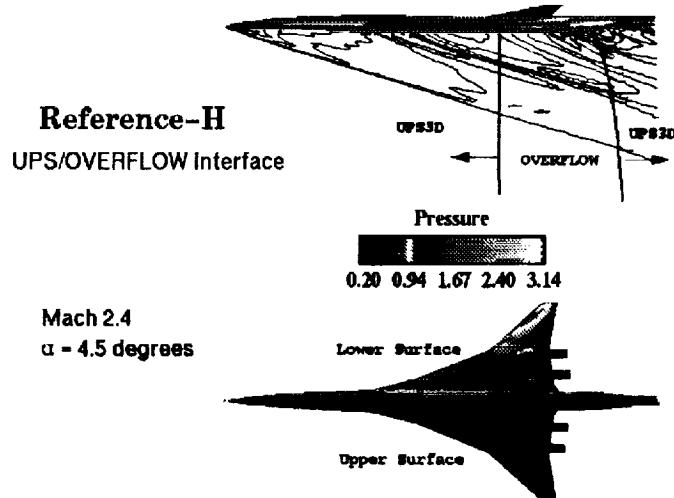
Research effort on *low-boom configuration* concept has been invested for the past four years. A new proposed route structure for HSCT's incorporating supersonic corridors over land and water has relaxed the sonic boom constraint somewhat. The objective of this study is twofold. First is to exercise the methodology of combining two different CFD codes to solve the near-field solution of a realistic HSCT configuration in an efficient and accurate manner. Second is to reduce the sonic boom loudness of a *performance configuration* concept, Reference-H, without jeopardizing the aerodynamic performance. The basic components of Reference-H are a fuselage, a pair of swept wings, and four nacelles.

##### 3.1.1 Reference-H Near-Field Study

The CFD codes used in this study are the UPS3D code and the OVERFLOW code. Both CFD codes has been described in Section 2.1.1 and 2.4, respectively. The former is an efficient space-marching code. However, it fails in the region where subsonic pocket exists; especially in the region of the wing/nacelle integration. The latter is a time-iterative code with Chimera overset grid concept, which makes the code more viable in solving the



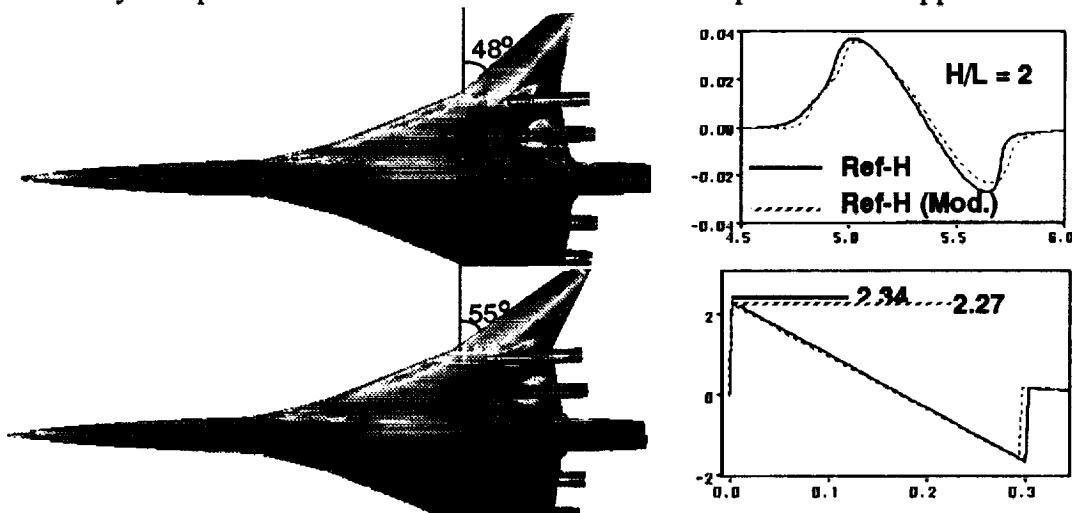
region of wing/nacelle integration. In this study, only inviscid flow is considered. Figure below summarizes the result of the CFD calculations.



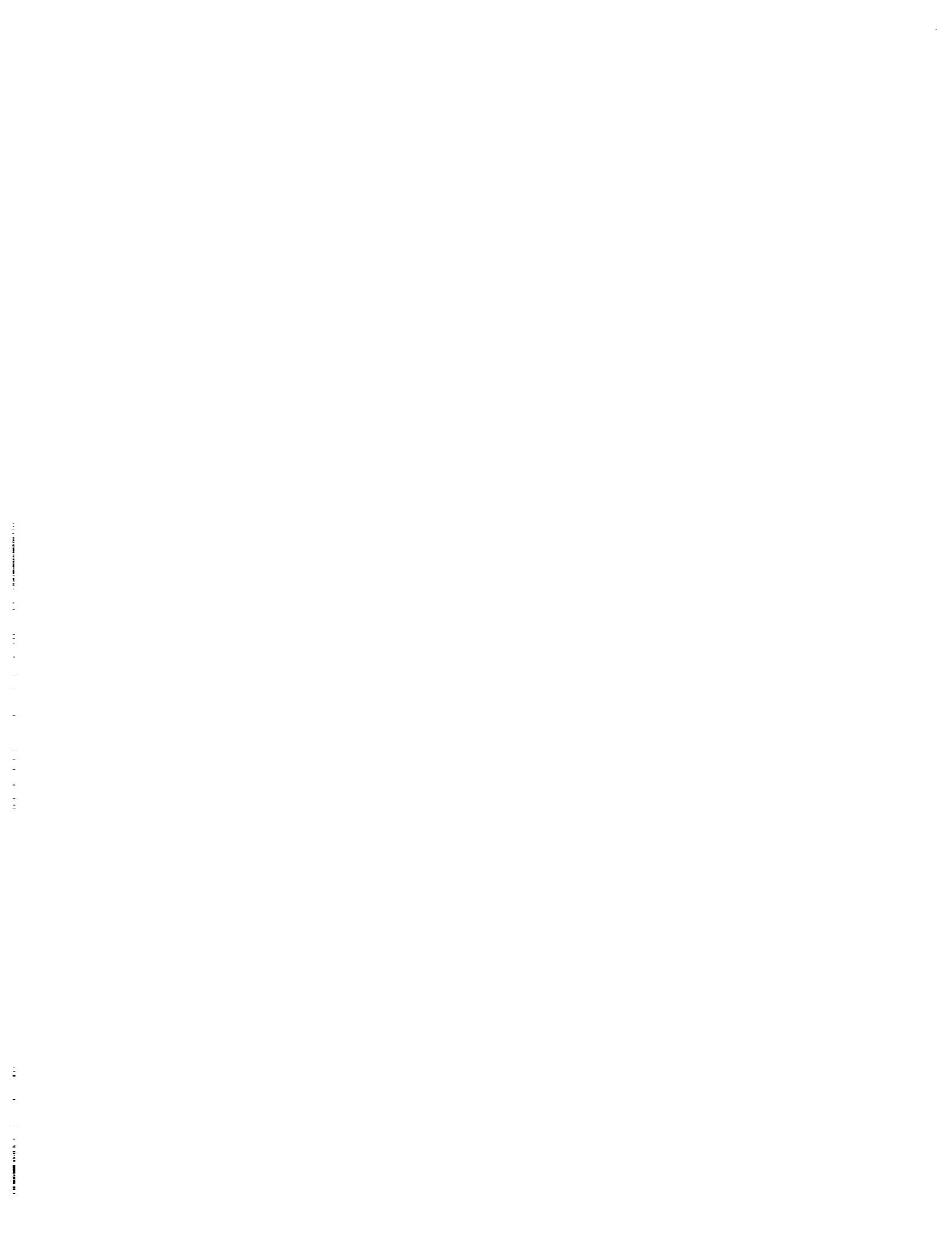
The near-field solution is studied for the case of Mach number 2.4 and angle of attack 4.5 degrees. Wind-tunnel data of the Reference-H validate the CFD method. Study shows that flow particles turn significantly over the outer nacelle compared with the inner nacelle. It indicates that the effect of the nacelle orientation might improve the aerodynamic performance.

### 3.1.2 Sonic Boom Softening

The sonic boom of the Reference-H configuration is also obtained. The calculation shows that the boom is an N-wave of 104 PLdB with 2.5 psf. bow shock on the ground. Details of the sonic boom prediction technique can be found in Ref. 10. Boom modification for performance aircraft is very much different from the low-boom aircraft for cruise Mach number and lift are higher. Therefore, the technique developed previously can not be strictly applied to Reference-H. However, changing the equivalent area can be helpful. The result of this study was presented in the 4th Sonic Boom Workshop.<sup>17</sup> Another approach to



reduce the boom is by experimenting the sweep angle. The figure above show one of the exercises done on the Ref-H. This exercise successfully shows Boeing how much boom

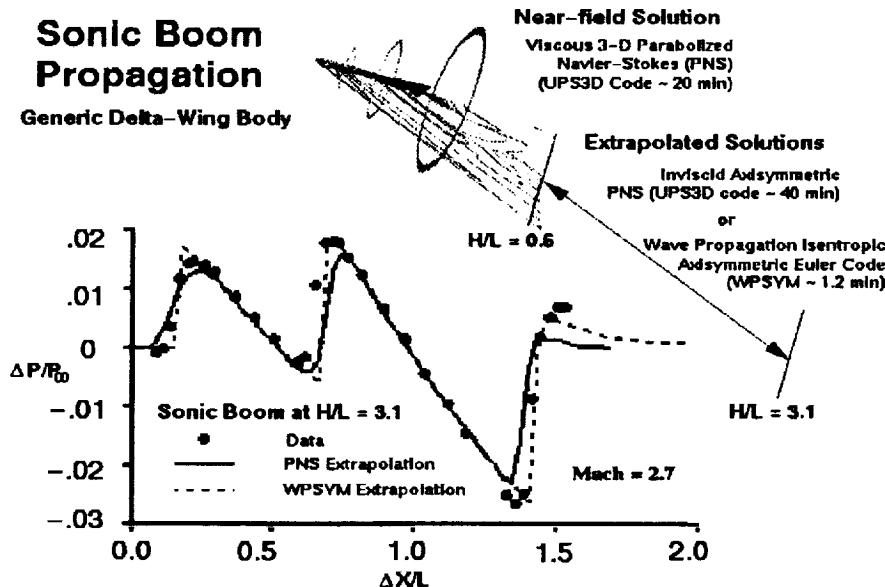


reduction can be achieved by redistributing the lift. A closer on-going technology communication with airframe industry is needed in order to achieve the goal of sonic boom softening on performance aircraft. A team consisting myself and other personnels from Boeing and NASA Langley has been formed to achieve the goal.

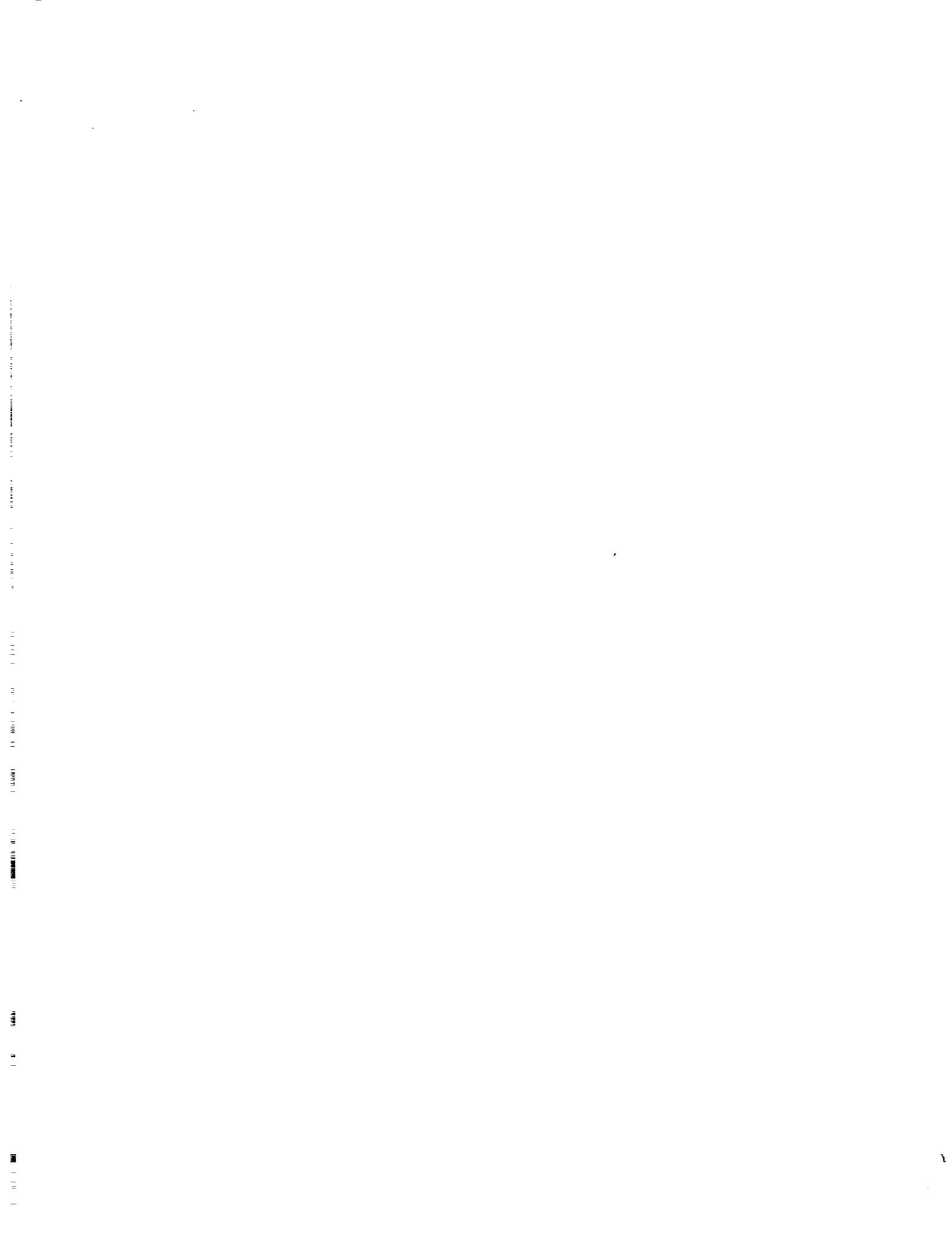
### 3.2 Sonic Boom Mid-Field Extrapolation (WPSYM)

In the beginning of 90's, sonic boom extrapolation technique was still relied on the linear theory developed in the 60's for the nonlinear techniques were computationally expensive. Today, a fast and accurate sonic boom extrapolation methodology is needed to bring the sonic boom extrapolation technique up to the 90's standard for HSCT design. The objective of this study is to develop an efficient and accurate higher-order computational method, solving the Euler equations, for supersonic aero-acoustic wave propagation.

An axisymmetric wave propagation code (WPSYM) has been developed for mid-field sonic boom extrapolation. This propagation code has been demonstrated as an efficient and accurate tool over the previous CFD method, described in Section 2.1.1 and Ref. 4, on a generic wing-body configuration. The figure below shows that a 3-D near-field solution



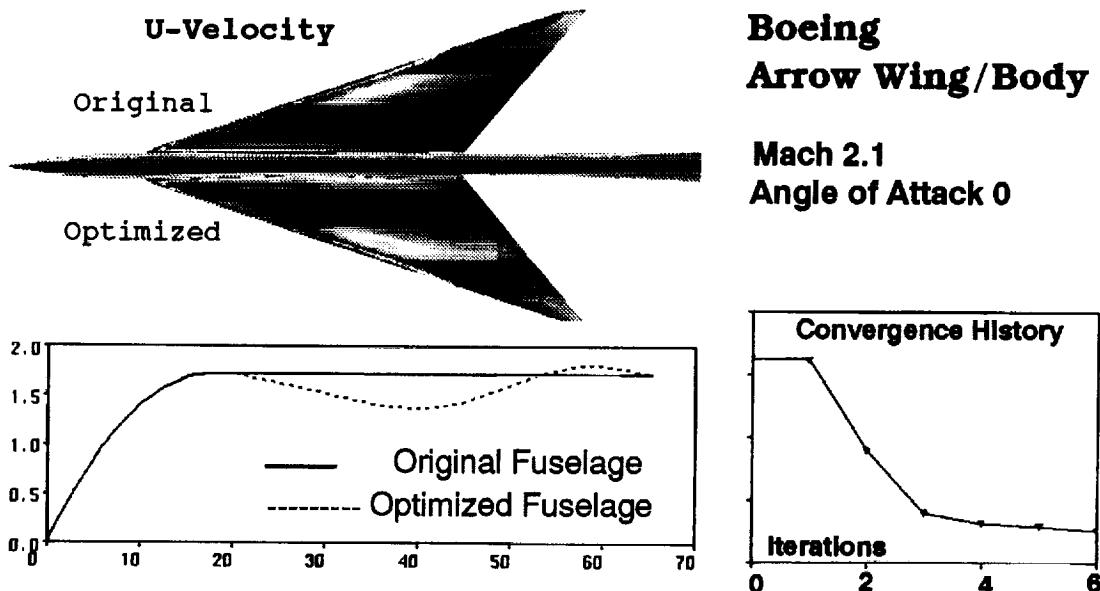
is obtained from UPS3D code; the result is then interfaced to two axisymmetric sonic boom extrapolation codes, namely, the axisymmetric version of UPS3D and the recent wave propagation code (WPSYM). The former takes 40 minutes on Cray-YMP, and the latter takes 1.2 minutes on the same machine. The x-y plot in the figure compares the numerical extrapolation results to wind-tunnel data. The result has been shown in NASA Technical Highlight and the methodology has been presented in the 4th Annual Sonic Boom Workshop at NASA Langley in June 1994.<sup>16</sup>



### 3.3 Optimizer on PVM (*IOWA*)

Moving to the world of parallel computing, the aerospace industry needs a numeric optimization tool in the parallel environment. One of the promising parallel computing concept is the network-based distributed computing. The Parallel Virtual Machine (PVM) is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. PVM allows users to link up engineering workstations to work as a single distributed-memory (parallel) machine. Merritt Smith and I wrote a manual on PVM for beginning users. A copy of the manual is attached in Appendix C.

A parallel optimizer based on nonlinear Quasi-Newton method has been developed and coupled with an efficient CFD code for basic aerodynamic design and study. This optimizer is called *IOWA* (parallel Optimizer With Aerodynamics). The figure below is a demonstration of *IOWA*. A Boeing arrow wing/body configuration is chosen in this

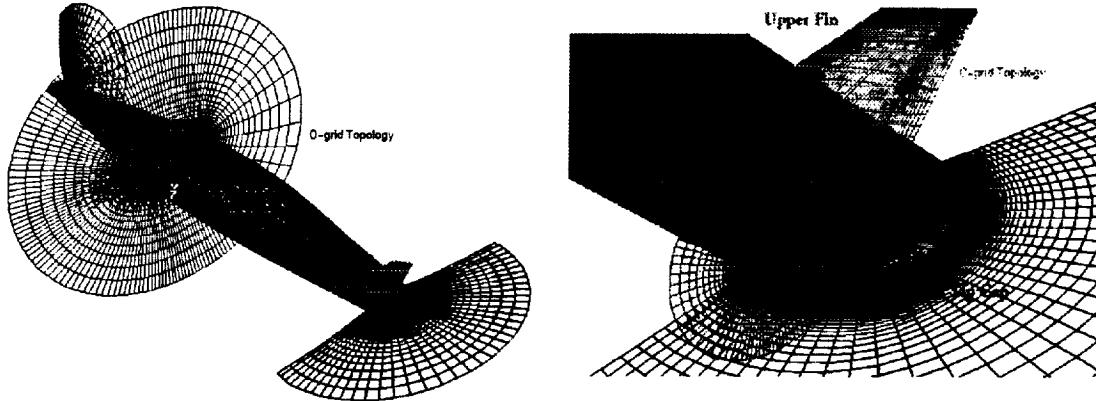


study. The fuselage radius is changed so that the wave drag is minimized. The parallel CFD optimization process takes 24 wall-clock hours on 4 SGI workstations to reduce the wave drag by 6.5%. The optimized result is a “coke bottle” shape fuselage, as expected by supersonic area rule. The convergence history of the optimization process is also shown in the figure. The optimizer is also coupled with a parallel CFD code, MEDUSA, to perform viscous 2-D multizone airfoil optimization supported by overset grid concept. The results will be presented at NASA CAS conference in March 1995.

### 3.4 Oblique All-Wing (OAW): CFD support

The OAW design team has asked for CFD support on the latest configuration OAW-3 from which a wind-tunnel model has been built and tested at Ames in June 1994. The figure below shows the chimera grid topology on the OAW-3 with fin. The design team want to compare the CFD result with the result from pressure sensitive paint (PSP). Therefore,





CFD calculations have to be done prior to the wind-tunnel test because color map from CFD result is need for PSP calibration.

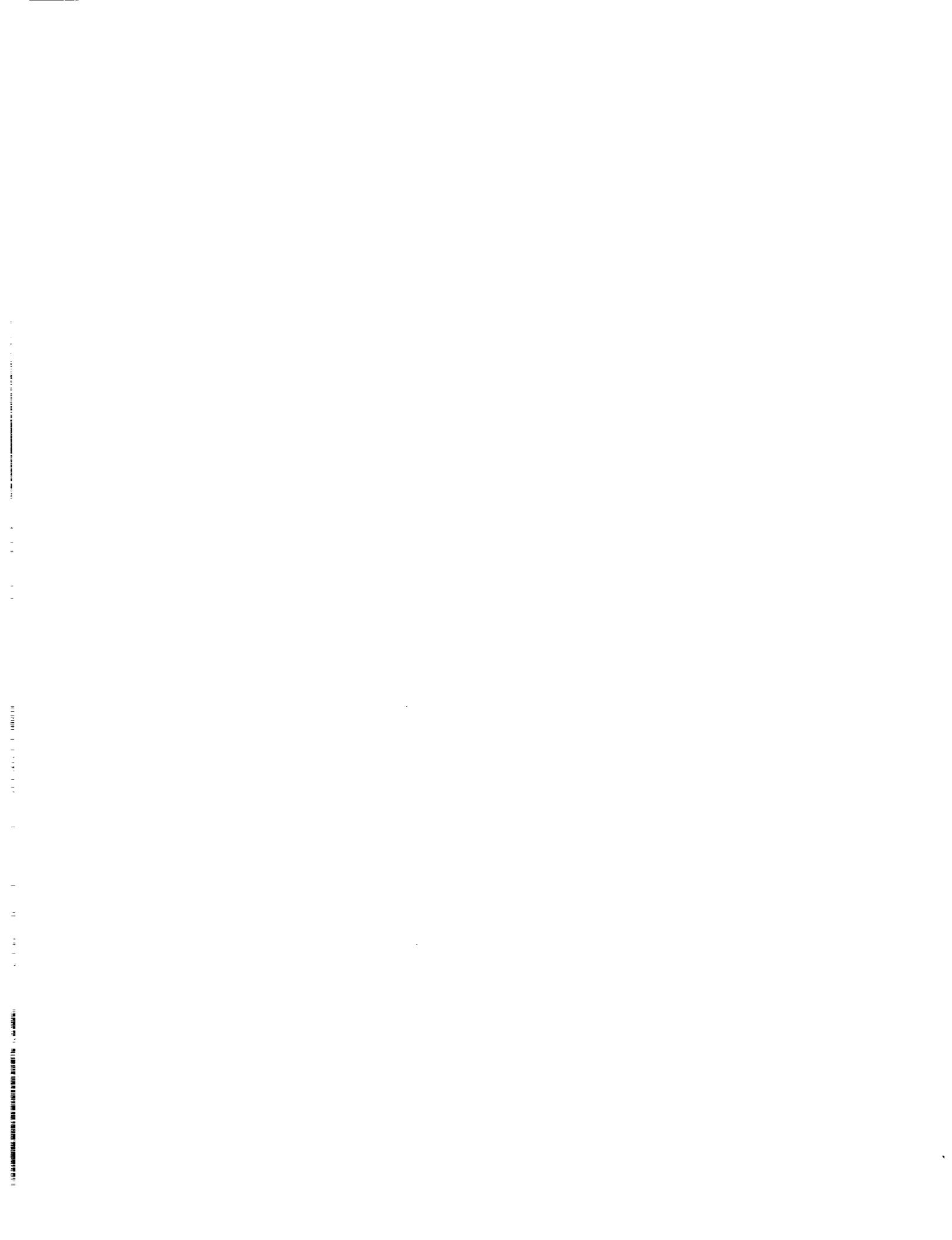
## 4 SUMMARY

The computational tools for sonic boom prediction, aerodynamic calculation, and configuration design of the current HSCT concept have been validated and applied to build wind-tunnel model for further testing and validation. The techniques developed in this five-year research and their applications, such as sonic boom prediction technique (Section 2.1), design of Ames Model 3 (Section 2.3) by CFD optimization (Section 2.2), and sonic boom softening for performance configuration (Section 3.1), have clearly shown support to the HSRP as it moved to its phase two period.

An accurate sonic boom extrapolation tool has always been an issue. It is because the flow phenomena in the atmosphere are nonlinear, but the common technique for extrapolation is linear acoustic theory developed in the 60's. On the other hand, CFD technique is too computationally expensive. Recently, a fast and accurate sonic boom extrapolation methodology (Section 3.2), solving the Euler equations for axisymmetric flow, has brought the sonic boom extrapolation technique up to the 90's standard.

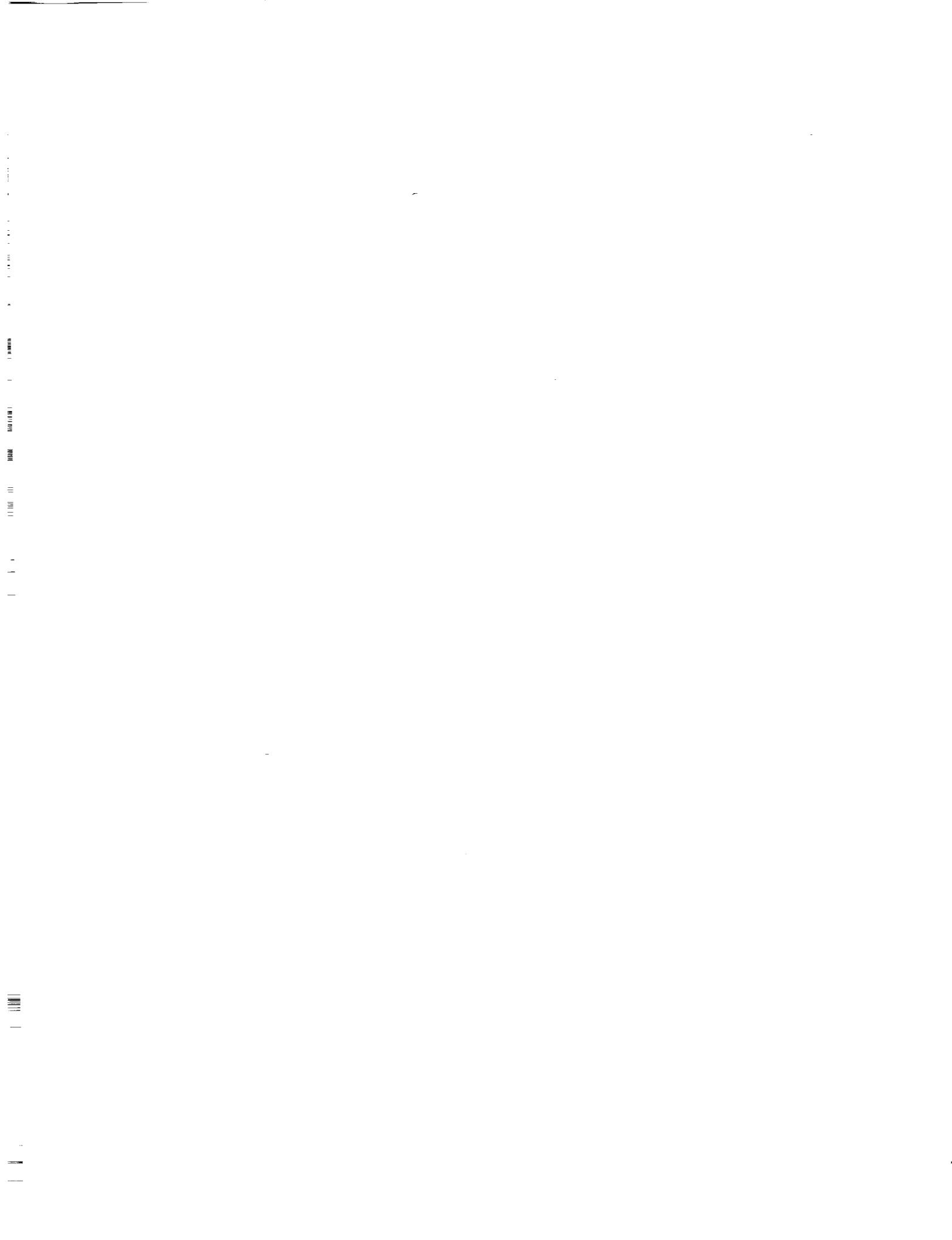
Parallel computing is a fast growing subject in the field of computer science because of the promising speed in number crunching computations. A new optimizer (Section 3.3) for parallel computing concept has been developed and tested for aerodynamic drag minimization. This optimizer is also coupled with a parallel CFD code so the whole optimization process is parallel. This is a promising method for CFD optimization making use of the computational resources of workstations, which unlike supercomputers spend most of their time idle.

Finally, the OAW concept is so attractive because of its overall performance in theory. In order to fully understand the concept, a wind-tunnel model is built. CFD Navier-Stokes calculations helps to identify the problem of the flow separation (Section 2.4), and also help to design the wing deflection for roll trim and alleviating the flow separation.



## 5 References

1. Lawrence, S., Chaussee, D., and Tannehill, J., "Application of an Upwind Algorithm to the 3-D Parabolized Navier-Stokes Equations," AIAA paper 87-1112, June 1987.
2. Whitham, G. B., "The flow Pattern of a Supersonic Projectile," Comm. Pure & Appl. Math., Vol. V, No. 3, 1952.
3. Lighthill, M. J., "Higher Approximation", General Theory of High Speed Aerodynamics, Vol. VI of High Speed Aerodynamics and Propulsion, Princeton University Press, 1954.
4. Cheung, S., Edwards, T., and Lawrence, S., "Application of CFD to Sonic Boom Near and Mid Flow-Field Prediction," J. of Aircraft, Vol. 29, No. 5, 1992.
5. Cheung, S., Edwards, T., and Lawrence, S., NASA TM 102867, August 1990.
6. Gill, P., Murray, W., Saunders, M., and Wright, M., Users Guide for NPSOL, Technical Report SOL 86-2, Stanford University, CA 94305, 1986.
7. Chan, W., and Steger, J., "A Generalized Scheme for Three-Dimension Hyperbolic Grid Generator," AIAA Paper 91-1588, 1991.
8. Heaslet, M. and Lomax, H., "Supersonic and Transonic Small Perturbation Theory," General Theory of High Speed Aerodynamics, Vol. VI of High Speed Aerodynamics and Propulsion, Princeton University Press, 1954.
9. Stevens, S. S., "Perceived Level of Noise by Mark VII and Decibels (E)," Jour. Acoust. Soc. Am., Vol. 51(2), 1972.
10. Cheung, S. and Edwards, T., "Supersonic Design Optimization Method for Aerodynamic Performance and Low Sonic Boom," High Speed Research: Sonic Boom, Vol. II, NACA CP-3173, 1992.
11. Haack, W., "Projectile Shapes for Smallest Wave Drag," Translation No. A9-T-3, Contract W33-038-ac-15004, ATI No. 27736, Air Material Command, US Air Force, Brown Univ., 1948.
12. Cheung, S., Aaronson, P., and Edwards, T., "CFD Optimization of a Theoretical Minimum Drag Body," AIAA paper 93-3421, August 1993.
13. Cheung, S., "Design Process of Ames Wind-Tunnel Model 3," High Speed Research: Sonic Boom, Vol. II, NACA CP-10133, 1993.
14. Water, M., Ardema, H., and Kroo, I., "Structure and Aerodynamic Considerations for an Oblique All-Wing Aircraft," AIAA paper 92-4220, August 1992.
15. Pulliam, T. and Chaussee, D., "A diagonal Form of an Implicit Approximation Factorization Algorithm," J. of Comp. Physics, Vol. 39, 1981, pp. 347-363.
16. Cheung, S., "Viscous CFD Analysis and Optimization of an Oblique All-Wing Transport," NASA CDCR-20005, November 1994.
17. Cheung, S., "Sonic Boom Softening of Reference-H," High Speed Research: Sonic Boom, NASA Langley, June 1994. (NASA CP will be published).



# Appendix A

## LHF (Fortran Listing)



UNIX™  
FORTRAN Program

SOURCE PROGRAM

LHF.f

DATE 7/14/94

TIME 5:00:11 pm

PAGE #

1

LINE #

SOURCE TEXT

```

1 C
2 C      PROGRAM LHF
3 C
4 C
5 C
6 C      This program calculates
7 C      1) the Lighthill F-function on body surface.
8 C          - a) input data
9 C          - b) design parameters
10 C
11 C      2) the overpressure signature at given distance R1.
12 C
13 C      3) the loudness level of the sonic boom at R1
14 C
15 C
16 C      INPUT -
17 C      lhf.in(3) : Input parameter
18 C      area.in(2) : Equivalent area distribution. (INAREA=0)
19 C      f.dat(4) : F-function distribution. (INAREA=2)
20 C      p.ro(8) : Pressure signature at near field distance R0. (R050)
21 C      coef.dat(33) : B-function due to lift. (LIFT=1)
22 C      grid.in(11) : Surface grid_PLOT3D plssar format. (INAREA=1)
23 C      Default case : Wing-body (INAREA=1);
24 C
25 C
26 C      OUTPUT -
27 C      xsa.out(12) : Equivalent area distribution and its derivative.
28 C      ffa.out(13) : F-functions on the body surface and at distance R1.
29 C      p.out(14) : Pressure signature at distance R1.
30 C      lcurve.F(34) : Integral curve of the shifted F-function.
31 C
32 C
33 C      Dr. Samson Cheung      [Tel: (415)-604-4462 ]
34 C      MCAT Institute
35 C      NASA Ames Research Center
36 C      M/S 258-01
37 C      Moffett Field, CA 94033
38 C      Date : 1992/3/4      Version 2.1
39 C
40 C
41 C      PARAMETER (KMAX=220,LMAX=1,JMAX=351,NMAX=900)
42 C
43 C      DIMENSION I(KMAX,LMAX,JMAX),Y(KMAX,LMAX,JMAX),Z(KMAX,LMAX,JMAX)
44 C      REAL R(NMAX), S(NMAX), SP(NMAX), TAU(NMAX), PTAU(NMAX)
45 C      COMMON/PAR/ PMACH,PFAC
46 C      LOGICAL WBDY
47 C      WBDY = .FALSE.
48 C
49 C      OPEN(UNIT=1, FILE='lhf.in', STATUS='OLD')
50 C
51 C      Read the input parameters
52 C      NAMELIST /PARA/ PMACH,PFAC,R0,R1,INAREA,LIFT,TORX
53 C      READ(1,PARA)
54 C      WRITE(6,PARA)
55 C
56 C
57 C      Input free-stream Mach number = PMACH
58 C      If TORX > 0, sonic boom varies time, else varies distance
59 C
60 C      Is the surface grid contains the whole configuration, or
61 C      only half-plane or only quarter-plane ?
62 C      PFAC = 1.      ! Whole plane
63 C      PFAC = 2.      ! Half-plane
64 C      PFAC = 4.      ! Quarter-plane
65 C
66 C      R1 will be the distance where the signature is captured.
67 C
68 C      If read in area distribution, INAREA = 0
69 C      read the grid           , INAREA = 1
70 C      the wing-body case     , INAREA =-1
71 C      read in F-function     , INAREA = 2
72 C      read the B-function    , LIFT = 1
73 C      read a signature at R0 , NO > 0.6
74 C
75 C
76 C      PI = 4.*ATAN(1.)
77 C      JDIM = JMAX
78 C      JDIM = 361
79 C
80 C
81 C      If R0 > 0, we read the pressure signature at R0 and extrapolate
82 C      IF(R0.GT.0.) GOTO 790
83 C
84 C      Find the area distribution of body configuration (sample case).
85 C      IF(INAREA.LT.0) THEN
86 C          CALL WBODY(JDIM,S,TAU)
87 C          WBDY = .TRUE.
88 C          CALL CONE(JDIM,S,TAU)
89 C          CALL MING(JDIM,S,TAU)
90 C          CALL SEARS(JDIM,S,TAU)
91 C          CALL BULLET(JDIM,S,TAU)
92 C          GOTO 270
93 C      ENDIF
94 C
95 C      Read in the given area distribution
96 C      IF(INAREA.EQ.0) THEN
97 C          OPEN(UNIT=2, FILE='area.in')
98 C          DO 50 J=1,NMAX
99 C              READ(2,*END=75) TAU(J),S(J)
100 C              S(J) = S(J)*PFAC
101 C 50 CONTINUE
102 C 75 CONTINUE
103 C  CLOSE(2)
104 C  JDIM = J-1
105 C  OPEN(UNIT=2, FILE='area.in')
106 C  DO 80 J=1,JDIM
107 C      WRITE(2,*) TAU(J),S(J)
108 C 80 CONTINUE
109 C  GOTO 270
110 C
111 C      ENDIF
112 C
113 C      Read in the F-function or define a F-function by calling FUNC
114 C      and integrate out the equivalent area by calling EAREA
115 C      IF(INAREA.EQ.2) THEN
116 C          CALL FUNC(TAU,FTAU,JDIM)
117 C          OPEN(UNIT=4, FILE='f.dat')
118 C          DO 100 J=1,NMAX
119 C              READ(4,*END=110) TAU(J),FTAU(J)
120 C 100 CONTINUE
120 C 110 CONTINUE

```

LINE #	SOURCE TEXT
121	c JDIM = J-1
122	CALL DISTARC(TAU,FTAU,J-1,TAU,FTAU,JDIM,10.,0)
123	CALL EAREA(S,FTAU,TAU,JDIM)
124	GOTO 270
125	ENDIF
126	C Read the 'PLOT3D surface grid file (Planar format)
127	and find the equivalent area distribution
128	OPEN(UNIT=11, FILE='grid.in',FORM='UNFORMATTED')
129	READ(11) JDIM,LDIM,JDIM
130	DO 200 J=1,JDIM
131	READ(11) ((X(K,L,J), K=1,JDIM),L=1,LDIM),
132	((Y(K,L,J), K=1,JDIM),L=1,LDIM),
133	((Z(K,L,J), K=1,JDIM),L=1,LDIM)
134	200 CONTINUE
135	C CLOSE(11)
136	C CALL EAREA(JDIM,LDIM,JDIM,X,Y,Z,KMAX,LMAX,JMAX,S)
137	C DO 220 J=1,JDIM
138	S(J) = PFAC*S(J)
139	TAU(J)=X(1,1,J)
140	220 CONTINUE
141	C 270 CONTINUE
142	C Obtain the derivative of the area distribution
143	C IF(LIFT.EQ.1) CALL BFUNC(JDIM,S,TAU)
144	C JDIMM1 = JDIM-1
145	DO 300 J=2,JDIMM1
146	A1=(TAU(J)-TAU(J+1))/((TAU(J)-TAU(J-1))*(TAU(J+1)-TAU(J-1)))
147	A2=(TAU(J)-TAU(J-1))/((TAU(J+1)-TAU(J))*(TAU(J+1)-TAU(J-1)))
148	SP(J) = A1*(S(J-1)-S(J))+A2*(S(J+1)-S(J))
149	300 CONTINUE
150	A2 = (TAU(3)-TAU(1))/((TAU(2)-TAU(1))*(TAU(3)-TAU(2)))
151	A3 = (TAU(2)-TAU(1))/((TAU(3)-TAU(2))*(TAU(3)-TAU(1)))
152	A1 = A2-A3
153	SP(1) = -A1*S(1)+A2*S(2)-A3*S(3)
154	A1 = (TAU(JDIM)-TAU(JDIM-2))
155	A2 = (TAU(JDIM)-TAU(JDIM-1))*(TAU(JDIM-1)-TAU(JDIM-2))
156	A3 = (TAU(JDIM)-TAU(JDIM-1))*(TAU(JDIM-1)-TAU(JDIM-2))
157	A0 = A1 - A2
158	SP(JDIM) = A2*S(JDIM-2)-A1*S(JDIM-1)+A0*S(JDIM)
159	c 1st order    SP(1) = (S(2)-S(1))/(TAU(2)-TAU(1))
160	c 1st order    SP(JDIM) = (S(JDIM)-S(JDIM-1))/(TAU(JDIM)-TAU(JDIM-1))
161	C
162	C Redistribute the S in equal spacing
163	CALL DISTARC(TAU,S,JDIM,TAU,S,JDIM,10.,0)
164	CALL DISTARC(TAU,SP,JDIM,TAU,SP,JDIM,10.,0)
165	DO 340 J=1,JDIM
166	R(J) = SQRT(S(J)/PI)
167	340 CONTINUE
168	C OPEN(UNIT=12,FILE='area.out')
169	WRITE(12,400)
170	400 FORMAT(37H#This is equivalent area distribution)
171	DO 450 J=1,JDIM
172	WRITE(12,580) TAU(J),S(J)
173	450 CONTINUE
174	WRITE(12,*)'
175	WRITE(12,451)
176	c 451 FORMAT(49H#This is the derivative of the area distribution)
177	DO 455 J=1,JDIM
178	WRITE(12,580) TAU(J),SP(J)
179	455 CONTINUE
180	C CLOSE(12)
181	C The interference between wing and body, sample case only.
182	IF(WBDY) CALL WB(JDIM,SP,TAU)
183	C
184	C Obtain the Lighthill F-function
185	CALL LIGHT1(TAU,R,SP,JDIM,FMACH,FTAU)
186	C
187	C Write out the Lighthill F-function at the body surface.
188	OPEN(UNIT=13,FILE='ffn.out')
189	WRITE(13,556)
190	556 FORMAT(49H#This is the Lighthill F-function at body surface)
191	DO 560 J=1,JDIM
192	WRITE(13,580) TAU(J),FTAU(J)
193	560 CONTINUE
194	580 FORMAT(2X,E16.8,1X,E16.8)
195	CLOSE(13)
196	790 CONTINUE
197	C
198	C Obtained the pressure signature at distance R1 from the body
199	CALL FFN(FTAU,TAU,FMACH,JDIM,R0,R1,TORX)
200	C
201	C CLOSE(3)
202	STOP
203	END

UNIX™  
FORTRAN ProgramSOURCE PROGRAM  
**LHF.f**DATE 7/14/94 PAGE #  
TIME 5:00:11 pm 3

LINE #

## SOURCE TEXT

```
220 SUBROUTINE LIGHT1(TAU,R,SP,N,FMACH,FTAU)
221 DIMENSION R(N),SP(N),TAU(N),FTAU(N)
222 C
223 PI= 4.*ATAN(1.)
224 BETA=SQRT(FMACH**2-1.0)
225 TAU(1)=0.
226 FTAU(1)=0.
227 C
228 DO 95 M=1,N
229   FTAU(M)=0.
230
231 DO 100 J=1,N
232   DO 102 I=2,N
233     IF(ABS(R(I)),LE.1.E-10) THEN
234       Z1 = 1.E+10
235       F4 = 0.
236       GOTO 98
237     ENDIF
238     AB=2.0/(BETA*R(I))
239     AB1=ABS(AB)
240     F1=SQRT(AB1)
241     F2=SP(I)-SP(I-1)
242     F3=F1+F2
243     F4=F3/(2.0*PI)
244     Z1=(TAU(J)-TAU(I))/(BETA*R(I))
245     XLO=-1.0
246     IF (Z1.LT.XLO) GO TO 96
247     IF (Z1.LT.4.0) GO TO 97
248     IF (Z1.GE.4.0) GO TO 98
249     HZ1=0.
250
251   FTAU(J)=FTAU(J)+HZ1+F4
252   GO TO 99
253   HZ1=.02937*Z1*Z1-.2175*Z1+.7531
254   FTAU(J)=FTAU(J)+HZ1+F4
255   GO TO 99
256   BB=1.0/(2.0*Z1)
257   BB1=ABS(BB)
258   HZ1=SQRT(BB)
259   FTAU(J)=FTAU(J)+HZ1+F4
260
261 99  CONTINUE
262 102 CONTINUE
263 100 CONTINUE
264 RETURN
265
266 END
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM  
LHF.f

DATE 7/14/94  
TIME 5:00:11 pm

4

SOURCE TEXT

LINE # SUBROUTINE EQUAREA(KDIM,LDIM,JDIM,X,Y,Z,KMAX,LMAX,JMAX,S)

```

265  SUBROUTINE EQUAREA(KDIM,LDIM,JDIM,X,Y,Z,KMAX,LMAX,JMAX,S)
266  C This subroutine finds the cross-section area of a surface grid
267  C which has symmetry plane at Y-axis. For each marching station
268  C (for each x) the area is found. The area is approximated by
269  C trapezoidal rule.
270  C
271  C      DIMENSION X(KMAX,LMAX,JMAX),Y(KMAX,LMAX,JMAX),Z(KMAX,LMAX,JMAX)
272  C      REAL S(JDIM)
273  C
274  C      DO 10 J=1,JDIM
275  C          DAREA = 0.
276  C          DO 5 K=2,KDIM
277  C              H = Y(K,LDIM,J)-Y(K-1,LDIM,J)
278  C              ADD = Z(K,LDIM,J)+Z(K-1,LDIM,J)
279  C              DAREA = DAREA + 0.5*ADD*H
280  C
281  C      CONTINUE
282  C
283  C      The unwanted base area:
284  C      H = Y(1,LDIM,J) - Y(KDIM,LDIM,J)
285  C      ADD = Z(1,LDIM,J)+Z(KDIM,LDIM,J)
286  C      BASE = ABS(0.5*ADD*H)
287  C
288  C      The area surrounded by half of the plane
289  C      S(J) = ABS(DAREA)-BASE
290  C
291  C 10  CONTINUE
292  C
293  C
294  C
295  C
296  C
297  C
298  C
299  C
300  C
301  C
302  C
303  C
304  C
305  C
306  C
307  C
308  C
309  C
310  C
311  C
312  C
313  C
314  C
315  C
316  C
317  C
318  C
319  C
320  C
321  C
322  C
323  C
324  C
325  C
326  C
327  C
328  C
329  C
330  C
331  C
332  C
333  C
334  C
335  C
336  C
337  C
338  C
339  C
340  C
341  C
342  C
343  C
344  C
345  C
346  C
347  C
348  C
349  C
350  C
351  C
352  C
353  C
354  C
355  C
356  C
357  C
358  C
359  C
360  C
361  C
362  C
363  C
364  C
365  C
366  C
367  C
368  C
369  C
370  C
371  C
372  C
373  C
374  C
375  C
376  C
377  C
378  C
379  C
380  C
381  C
382  C
383  C
384  C
385  C
386  C
387  C
388  C
389  C
390  C
391  C
392  C
393  C
394  C
395  C
396  C
397  C
398  C
399  C
400  C
401  C
402  C
403  C
404  C
405  C
406  C
407  C
408  C
409  C
410  C
411  C
412  C
413  C
414  C
415  C
416  C
417  C
418  C
419  C
420  C
421  C
422  C
423  C
424  C
425  C
426  C
427  C
428  C
429  C
430  C
431  C
432  C
433  C
434  C
435  C
436  C
437  C
438  C
439  C
440  C
441  C
442  C
443  C
444  C
445  C
446  C
447  C
448  C
449  C
450  C
451  C
452  C
453  C
454  C
455  C
456  C
457  C
458  C
459  C
460  C
461  C
462  C
463  C
464  C
465  C
466  C
467  C
468  C
469  C
470  C
471  C
472  C
473  C
474  C
475  C
476  C
477  C
478  C
479  C
480  C
481  C
482  C
483  C
484  C
485  C
486  C
487  C
488  C
489  C
490  C
491  C
492  C
493  C
494  C
495  C
496  C
497  C
498  C
499  C
500  C
501  C
502  C
503  C
504  C
505  C
506  C
507  C
508  C
509  C
510  C
511  C
512  C
513  C
514  C
515  C
516  C
517  C
518  C
519  C
520  C
521  C
522  C
523  C
524  C
525  C
526  C
527  C
528  C
529  C
530  C
531  C
532  C
533  C
534  C
535  C
536  C
537  C
538  C
539  C
540  C
541  C
542  C
543  C
544  C
545  C
546  C
547  C
548  C
549  C
550  C
551  C
552  C
553  C
554  C
555  C
556  C
557  C
558  C
559  C
560  C
561  C
562  C
563  C
564  C
565  C
566  C
567  C
568  C
569  C
570  C
571  C
572  C
573  C
574  C
575  C
576  C
577  C
578  C
579  C
580  C
581  C
582  C
583  C
584  C
585  C
586  C
587  C
588  C
589  C
590  C
591  C
592  C
593  C
594  C
595  C
596  C
597  C
598  C
599  C
600  C
601  C
602  C
603  C
604  C
605  C
606  C
607  C
608  C
609  C
610  C
611  C
612  C
613  C
614  C
615  C
616  C
617  C
618  C
619  C
620  C
621  C
622  C
623  C
624  C
625  C
626  C
627  C
628  C
629  C
630  C
631  C
632  C
633  C
634  C
635  C
636  C
637  C
638  C
639  C
640  C
641  C
642  C
643  C
644  C
645  C
646  C
647  C
648  C
649  C
650  C
651  C
652  C
653  C
654  C
655  C
656  C
657  C
658  C
659  C
660  C
661  C
662  C
663  C
664  C
665  C
666  C
667  C
668  C
669  C
670  C
671  C
672  C
673  C
674  C
675  C
676  C
677  C
678  C
679  C
680  C
681  C
682  C
683  C
684  C
685  C
686  C
687  C
688  C
689  C
690  C
691  C
692  C
693  C
694  C
695  C
696  C
697  C
698  C
699  C
700  C
701  C
702  C
703  C
704  C
705  C
706  C
707  C
708  C
709  C
710  C
711  C
712  C
713  C
714  C
715  C
716  C
717  C
718  C
719  C
720  C
721  C
722  C
723  C
724  C
725  C
726  C
727  C
728  C
729  C
730  C
731  C
732  C
733  C
734  C
735  C
736  C
737  C
738  C
739  C
740  C
741  C
742  C
743  C
744  C
745  C
746  C
747  C
748  C
749  C
750  C
751  C
752  C
753  C
754  C
755  C
756  C
757  C
758  C
759  C
760  C
761  C
762  C
763  C
764  C
765  C
766  C
767  C
768  C
769  C
770  C
771  C
772  C
773  C
774  C
775  C
776  C
777  C
778  C
779  C
780  C
781  C
782  C
783  C
784  C
785  C
786  C
787  C
788  C
789  C
790  C
791  C
792  C
793  C
794  C
795  C
796  C
797  C
798  C
799  C
800  C
801  C
802  C
803  C
804  C
805  C
806  C
807  C
808  C
809  C
810  C
811  C
812  C
813  C
814  C
815  C
816  C
817  C
818  C
819  C
820  C
821  C
822  C
823  C
824  C
825  C
826  C
827  C
828  C
829  C
830  C
831  C
832  C
833  C
834  C
835  C
836  C
837  C
838  C
839  C
840  C
841  C
842  C
843  C
844  C
845  C
846  C
847  C
848  C
849  C
850  C
851  C
852  C
853  C
854  C
855  C
856  C
857  C
858  C
859  C
860  C
861  C
862  C
863  C
864  C
865  C
866  C
867  C
868  C
869  C
870  C
871  C
872  C
873  C
874  C
875  C
876  C
877  C
878  C
879  C
880  C
881  C
882  C
883  C
884  C
885  C
886  C
887  C
888  C
889  C
890  C
891  C
892  C
893  C
894  C
895  C
896  C
897  C
898  C
899  C
900  C
901  C
902  C
903  C
904  C
905  C
906  C
907  C
908  C
909  C
910  C
911  C
912  C
913  C
914  C
915  C
916  C
917  C
918  C
919  C
920  C
921  C
922  C
923  C
924  C
925  C
926  C
927  C
928  C
929  C
930  C
931  C
932  C
933  C
934  C
935  C
936  C
937  C
938  C
939  C
940  C
941  C
942  C
943  C
944  C
945  C
946  C
947  C
948  C
949  C
950  C
951  C
952  C
953  C
954  C
955  C
956  C
957  C
958  C
959  C
960  C
961  C
962  C
963  C
964  C
965  C
966  C
967  C
968  C
969  C
970  C
971  C
972  C
973  C
974  C
975  C
976  C
977  C
978  C
979  C
980  C
981  C
982  C
983  C
984  C
985  C
986  C
987  C
988  C
989  C
990  C
991  C
992  C
993  C
994  C
995  C
996  C
997  C
998  C
999  C
1000  C
1001  C
1002  C
1003  C
1004  C
1005  C
1006  C
1007  C
1008  C
1009  C
1010  C
1011  C
1012  C
1013  C
1014  C
1015  C
1016  C
1017  C
1018  C
1019  C
1020  C
1021  C
1022  C
1023  C
1024  C
1025  C
1026  C
1027  C
1028  C
1029  C
1030  C
1031  C
1032  C
1033  C
1034  C
1035  C
1036  C
1037  C
1038  C
1039  C
1040  C
1041  C
1042  C
1043  C
1044  C
1045  C
1046  C
1047  C
1048  C
1049  C
1050  C
1051  C
1052  C
1053  C
1054  C
1055  C
1056  C
1057  C
1058  C
1059  C
1060  C
1061  C
1062  C
1063  C
1064  C
1065  C
1066  C
1067  C
1068  C
1069  C
1070  C
1071  C
1072  C
1073  C
1074  C
1075  C
1076  C
1077  C
1078  C
1079  C
1080  C
1081  C
1082  C
1083  C
1084  C
1085  C
1086  C
1087  C
1088  C
1089  C
1090  C
1091  C
1092  C
1093  C
1094  C
1095  C
1096  C
1097  C
1098  C
1099  C
1100  C
1101  C
1102  C
1103  C
1104  C
1105  C
1106  C
1107  C
1108  C
1109  C
1110  C
1111  C
1112  C
1113  C
1114  C
1115  C
1116  C
1117  C
1118  C
1119  C
1120  C
1121  C
1122  C
1123  C
1124  C
1125  C
1126  C
1127  C
1128  C
1129  C
1130  C
1131  C
1132  C
1133  C
1134  C
1135  C
1136  C
1137  C
1138  C
1139  C
1140  C
1141  C
1142  C
1143  C
1144  C
1145  C
1146  C
1147  C
1148  C
1149  C
1150  C
1151  C
1152  C
1153  C
1154  C
1155  C
1156  C
1157  C
1158  C
1159  C
1160  C
1161  C
1162  C
1163  C
1164  C
1165  C
1166  C
1167  C
1168  C
1169  C
1170  C
1171  C
1172  C
1173  C
1174  C
1175  C
1176  C
1177  C
1178  C
1179  C
1180  C
1181  C
1182  C
1183  C
1184  C
1185  C
1186  C
1187  C
1188  C
1189  C
1190  C
1191  C
1192  C
1193  C
1194  C
1195  C
1196  C
1197  C
1198  C
1199  C
1200  C
1201  C
1202  C
1203  C
1204  C
1205  C
1206  C
1207  C
1208  C
1209  C
1210  C
1211  C
1212  C
1213  C
1214  C
1215  C
1216  C
1217  C
1218  C
1219  C
1220  C
1221  C
1222  C
1223  C
1224  C
1225  C
1226  C
1227  C
1228  C
1229  C
1230  C
1231  C
1232  C
1233  C
1234  C
1235  C
1236  C
1237  C
1238  C
1239  C
1240  C
1241  C
1242  C
1243  C
1244  C
1245  C
1246  C
1247  C
1248  C
1249  C
1250  C
1251  C
1252  C
1253  C
1254  C
1255  C
1256  C
1257  C
1258  C
1259  C
1260  C
1261  C
1262  C
1263  C
1264  C
1265  C
1266  C
1267  C
1268  C
1269  C
1270  C
1271  C
1272  C
1273  C
1274  C
1275  C
1276  C
1277  C
1278  C
1279  C
1280  C
1281  C
1282  C
1283  C
1284  C
1285  C
1286  C
1287  C
1288  C
1289  C
1290  C
1291  C
1292  C
1293  C
1294  C
1295  C
1296  C
1297  C
1298  C
1299  C
1300  C
1301  C
1302  C
1303  C
1304  C
1305  C
1306  C
1307  C
1308  C
1309  C
1310  C
1311  C
1312  C
1313  C
1314  C
1315  C
1316  C
1317  C
1318  C
1319  C
1320  C
1321  C
1322  C
1323  C
1324  C
1325  C
1326  C
1327  C
1328  C
1329  C
1330  C
1331  C
1332  C
1333  C
1334  C
1335  C
1336  C
1337  C
1338  C
1339  C
1340  C
1341  C
1342  C
1343  C
1344  C
1345  C
1346  C
1347  C
1348  C
1349  C
1350  C
1351  C
1352  C
1353  C
1354  C
1355  C
1356  C
1357  C
1358  C
1359  C
1360  C
1361  C
1362  C
1363  C
1364  C
1365  C
1366  C
1367  C
1368  C
1369  C
1370  C
1371  C
1372  C
1373  C
1374  C
1375  C
1376  C
1377  C
1378  C
1379  C
1380  C
1381  C
1382  C
1383  C
1384  C
1385  C
1386  C
1387  C
1388  C
1389  C
1390  C
1391  C
1392  C
1393  C
1394  C
1395  C
1396  C
1397  C
1398  C
1399  C
1400  C
1401  C
1402  C
1403  C
1404  C
1405  C
1406  C
1407  C
1408  C
1409  C
1410  C
1411  C
1412  C
1413  C
1414  C
1415  C
1416  C
1417  C
1418  C
1419  C
1420  C
1421  C
1422  C
1423  C
1424  C
1425  C
1426  C
1427  C
1428  C
1429  C
1430  C
1431  C
1432  C
1433  C
1434  C
1435  C
1436  C
1437  C
1438  C
1439  C
1440  C
1441  C
1442  C
1443  C
1444  C
1445  C
1446  C
1447  C
1448  C
1449  C
1450  C
1451  C
1452  C
1453  C
1454  C
1455  C
1456  C
1457  C
1458  C
1459  C
1460  C
1461  C
1462  C
1463  C
1464  C
1465  C
1466  C
1467  C
1468  C
1469  C
1470  C
1471  C
1472  C
1473  C
1474  C
1475  C
1476  C
1477  C
1478  C
1479  C
1480  C
1481  C
1482  C
1483  C
1484  C
1485  C
1486  C
1487  C
1488  C
1489  C
1490  C
1491  C
1492  C
1493  C
1494  C
1495  C
1496  C
1497  C
1498  C
1499  C
1500  C
1501  C
1502  C
1503  C
1504  C
1505  C
1506  C
1507  C
1508  C
1509  C
1510  C
1511  C
1512  C
1513  C
1514  C
1515  C
1516  C
1517  C
1518  C
1519  C
1520  C
1521  C
1522  C
1523  C
1524  C
1525  C
1526  C
1527  C
1528  C
1529  C
1530  C
1531  C
1532  C
1533  C
1534  C
1535  C
1536  C
1537  C
1538  C
1539  C
1540  C
1541  C
1542  C
1543  C
1544  C
1545  C
1546  C
1547  C
1548  C
1549  C
1550  C
1551  C
1552  C
1553  C
1554  C
1555  C
1556  C
1557  C
1558  C
1559  C
1560  C
1561  C
1562  C
1563  C
1564  C
1565  C
1566  C
1567  C
1568  C
1569  C
1570  C
1571  C
1572  C
1573  C
1574  C
1575  C
1576  C
1577  C
1578  C
1579  C
1580  C
1581  C
1582  C
1583  C
1584  C
1585  C
1586  C
1587  C
1588  C
1589  C
1590  C
1591  C
1592  C
1593  C
1594  C
1595  C
1596  C
1597  C
1598  C
1599  C
1600  C
1601  C
1602  C
1603  C
1604  C
1605  C
1606  C
1607  C
1608  C
1609  C
1610  C
1611  C
1612  C
1613  C
1614  C
1615  C
1616  C
1617  C
1618  C
1619  C
1620  C
1621  C
1622  C
1623  C
1624  C
1625  C
1626  C
1627  C
1628  C
1629  C
1630  C
1631  C
1632  C
1633  C
1634  C
1635  C
1636  C
1637  C
1638  C
1639  C
1640  C
1641  C
1642  C
1643  C
1644  C
1645  C
1646  C
1647  C
1648  C
1649  C
1650  C
1651  C
1652  C
1653  C
1654  C
1655  C
1656  C
1657  C
1658  C
1659  C
1660  C
1661  C
1662  C
1663  C
1664  C
1665  C
1666  C
1667  C
1668  C
1669  C
1670  C
1671  C
1672  C
1673  C
1674  C
1675  C
1676  C
1677  C
1678  C
1679  C
1680  C
1681  C
1682  C
1683  C
1684  C
1685  C
1686  C
1687  C
1688  C
1689  C
1690  C
1691  C
1692  C
1693  C
1694  C
1695  C
1696  C
1697  C
1698  C
1699  C
1700  C
1701  C
1702  C
1703  C
1704  C
1705  C
1706  C
1707  C
1708  C
1709  C
1710  C
1711  C
1712  C
1713  C
1714  C
1715  C
1716  C
1717  C
1718  C
1719  C
1720  C
1721  C
1722  C
1723  C
1724  C
1725  C
1726  C
1727  C
1728  C
1729  C
1730  C
1731  C
1732  C
1733  C
1734  C
1735  C
1736  C
1737  C
1738  C
1739  C
1740  C
1741  C
1742  C
1743  C
1744  C
1745  C
1746  C
1747  C
1748  C
1749  C
1750  C
1751  C
1752  C
1753  C
1754  C
1755  C
1756  C
1757  C
1758  C
1759  C
1760  C
1761  C
1762  C
1763  C
1764  C
1765  C
1766  C
1767  C
1768  C
1769  C
1770  C
1771  C
1772  C
1773  C
1774  C
1775  C
1776  C
1777  C
1778  C
1779  C
1780  C
1781  C
1782  C
1783  C
1784  C
1785  C
1786  C
1787  C
1788  C
1789  C
1790  C
1791  C
1792  C
1793  C
1794  C
1795  C
1796  C
1797  C
1798  C
1799  C
1800  C
1801  C
1802  C
1803  C
1804  C
1805  C
1806  C
1807  C
1808  C
1809  C
1810  C
1811  C
1812  C
1813  C
1814  C
1815  C
1816  C
1817  C
1818  C
1819  C
1820  C
1821  C
1822  C
1823  C
1824  C
1825  C
1826  C
1827  C
1828  C
1829  C
1830  C
1831  C
1832  C
1833  C
1834  C
1835  C
1836  C
1837  C
1838  C
1839  C
1840  C
1841  C
1842  C
1843  C
1844  C
1845  C
1846  C
1847  C
1848  C
1849  C
1850  C
1851  C
1852  C
1853  C
1854  C
1855  C
1856  C
1857  C
1858  C
1859  C
1860  C
1861  C
1862  C
1863  C
1864  C
1865  C
1866  C
1867  C
1868  C
1869  C
1870  C
1871  C
1872  C
1873  C
1874  C
1875  C
1876  C
1877  C
1878  C
1879  C
1880  C
1881  C
1882  C
1883  C
1884  C
1885  C
1886  C
1887  C
1888  C
1889  C
1890  C
1891  C
1892  C
1893  C
1894  C
1895  C
1896  C
1897  C
1898  C
1899  C
1900  C
1901  C
1902  C
1903  C
1904  C
1905  C
1906  C
1907  C
1908  C
1909  C
1910  C
1911  C
1912  C
1913  C
1914  C
1915  C
1916  C
1917  C
1918  C
1919  C
192
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM

LHF.f

DATE

7/14/94

PAGE #

5

LINE #

SOURCE TEXT

```

294      SUBROUTINE FFN(F,X,FNACH,NP,R0,R1,TORX)
295
296      C This program uses F-function theory to predict the pressure
297      C signature at far field when an initial pressure signature is given
298      C
299      PARAMETER (NMAX=1400)
300      DIMENSION F(NP),X(NP),Y(NMAX),P(NMAX)
301      DIMENSION YSTP(NMAX)
302      DIMENSION DBLVL(3)
303
304      OPEN(UNIT=14,FILE='p.out')
305
306      C Input of initial parameters and pressure signal
307
308      FMACH = Free-Stream Mach number
309      R0   = Initial distance from the body (altitude)
310      R1   = Final distance from the body (altitude)
311      NP   = Number of data (NP < NMAX)
312      NMAX = This should be large enough to resolve the signature
313      TORX > 0, sonic boom versus time, else versus distance
314
315      NAMELIST /XPSCALE/ XSCALE,PSCALE,Ag,Pg,Pa,IRISE,ALT
316
317      C Read the input parameters
318      READ(3,XPSCALE)
319      WRITE(6,XPSCALE)
320
321      C Define the parameters used in the F-function theory
322
323      GAMMA = 1.4
324      B = SQRT(FMACH**2 - 1.)
325      CAP = (GAMMA+1.)*FMACH**4/(SQRT(2.)*B**1.5)
326      SRRI = SQRT(R1)
327
328
329      C If R0 > 0, extrapolate from R0 to R1. First calculate the F-in.
330      IF(R0.GT.0.) THEN
331          OPEN(UNIT=8, FILE='p.r0')
332          DO 15 I=1,NMAX
333              READ(8,*END=30)I(I),P(I)
334          15 CONTINUE
335          CONTINUE
336          CLOSE(8)
337          NP = I-1
338          SRRO = SQRT(R0)
339          DO 35 I=1,NP
340              F(I) = SQRT(2.*B*R0)*P(I)/(GAMMA*FMACH*FMACH)
341              X(I) = X(I) - B*R0 + CAP*SRRO*F(I)
342          35 CONTINUE
343          CONTINUE
344          ENDIF
345
346      C Y is transposed coordinate
347
348      C      WRITE(13,49)
349      DO 50 I=1,NP
350          Y(I) = X(I) - CAP*SRRI*F(I)
351          WRITE(13,'(43H    )') Y(I),F(I)
352      50 CONTINUE
353
354      C Find the largest and smallest values of Y
355
356      YMAX = -1.E+8
357      YMIN = 1.E+8
358      DO 55 I=1,NP
359          IF(Y(I) .GE. YMAX) YMAX=Y(I)
360          IF(Y(I) .LE. YMIN) YMIN=Y(I)
361      55 CONTINUE
362
363
364      C Print out the integral curve of the shifted F-function
365      CALL INTF(NP,Y,F)
366
367      C Need to march in Y-direction, define the step
368
369      YSTP(1) = YMIN
370      YDIS = YMAX-YMIN
371      DY = YDIS/FLOAT(NMAX-1)
372      DO 80 J=2,NMAX
373          YSTP(J) = YSTP(J-1) + DY
374      80 CONTINUE
375
376      C March through the shifted F-function, check area-balance and
377      C place the shock.
378      CALL MARCH(NMAX,NP,Y,YSTP,F)
379
380      C Obtain the solutions
381      NOTE: If TORX>0, the sonic boom is in the form (P-Pinf) vs time
382      or it is in the form (P-Pinf)/Pinf vs distance.
383      DO 150 I=1,NP
384          P(I) = GAMMA*FMACH*FMACH*F(I)/SQRT(2.*B*R1)
385          X(I) = Y(I) + B*R1
386      150 CONTINUE
387
388
389      C Make the data points in evenly distributed manner and
390      C scale the sonic if desired
391      DO 180 I=1,NP
392          X(I) = X(I)*XSCALE
393          P(I) = P(I)*PSCALE
394      180 CONTINUE
395
396      C Atmospheric aspect
397      ALT = Altitude
398      Ag = speed of sound at ground in ft/sec
399      P0 = reference pressure lb/ft^2 = SQRT(Pa*Pg)
400
401      Pg = pressure at the ground
402      Pa = pressure at flight altitude
403      PO = SQRT(Pg*Pa)
404      VEL = FMACH*Ag
405      TREF = X(I)/VEL
406      IF(TORX.GT.0.) THEN
407          DO 260 I=1,NP
408              X(I) = X(I)/VEL - TREF
409              P(I) = P(I)*PO
410          260 CONTINUE
411
412      C The signal (DP vs Time) is calculated, use a empirical program to
413      C calculate the rise time, and embed the rise time into the signature.

```

LINE #

SOURCE TEXT

```
414 C Note: Unit used is still the stupid English unit!
415 C CALL RISETIME(FMACH,P,X,NP,ALT,IRISE)
416 C
417 C Obtain the noise level
418 C CALL NOISE(DBLVL,X,P,NP)
419 C
420 C Write the dB(PL) value out
421 C WRITE(14,500)DBLVL(1),DBLVL(2),DBLVL(3)
422 C
500 FORMAT
423   ('!Noise level ',F10.4,'PLdB',3X,F10.4,'dB(A)',3X,F10.4,'dB(C)')
424
425
426 C ... WRITE THE SONIC BOOM ...
427 C
428 555 FORMAT(28H#The pressure signal at R1= ,F10.4)
429 DO 670 I=1,NP
430   C
431   670 CONTINUE
432   C
433   700 FORMAT(3X,E20.8,2X,E16.6)
434 C
435 C CLOSE(14)
436 C
437 C RETURN
438 C
439 C END
```

## LINE #

## SOURCE TEXT

```
438      SUBROUTINE INTF(NP,Y,F)
439      C This program print out the integral curve of the shifted F-function
440      DIMENSION F(NP),Y(NP)
441      OPEN(UNIT=34,FILE='icurve_F',FORM='FORMATTED')
442      C
443      SUMF = 0
444      WRITE(34,120)
445      DO 100 J=2,np
446      DY = Y(J)-Y(J-1)
447      SUMF = SUMF + 0.5*DY*(F(J)+F(J-1))
448      WRITE(34,130)Y(J),SUMF
449 100  CONTINUE
450 120  FORMAT(42H# Integral curve of the shifted F-function)
451 130  FORMAT(2B16.6)
452      C
453      CLOSE(34)
454      RETURN
455      END
```

LINE #	SOURCE TEXT
457	SUBROUTINE SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,III)
458	DIMENSION F(NP),Y(NP)
459	DIMENSION YSTP(NMAX)
460	DIMENSION INDEX(40),FS(40)
461	COMMON/SHOCK/ INSLCT
462	
463	YEND = Y(NP)
464	FIRST = 1.
465	DO 500 J=2,NMAX
466	YS = YSTP(J)
467	C Get the points on the curve for integration, start searching from
468	IS to IE
469	C
470	CALL POINT(NP,Y,F,INDEX,FS,YS,INSLCT)
471	C After obtain the integration points, we can integrate and
472	find the Area
473	C
474	IF(INSLCT.GT.2) THEN
475	IF(III.EQ.3) INSLCT = 3
476	IS=INDEX(1)
477	IE=INDEX(INSLCT)
478	CALL AREA(NP,Y,F,YS,FS,IS,IE,IFLAT2)
479	ELSE
480	The tail shock is already formed, leave program
481	IF(INSLCT.LE.1 .AND. YS.GT.YEND-1.05) RETURN
482	FIRST=1.
483	GOTO 500
484	ENDIF
485	C
486	IF(FIRST.GT.0.) IFLAT1 = IFLAT2
487	IF(IFLAT2.EQ.0) RETURN
488	FIRST = -1.
489	C
490	IF IFLAT = 0, YS is the point that have area balanced.
491	If IFLAT2 and IFLAT1 are in different sign, i.e.,
492	the correct point should be between I and I-1.
493	Use bisection method to fine the correct point Y(ISTART)
494	C
495	IF(IFLAT1+IFLAT2 .LT. 0.) THEN
496	Y1 = YSTP(J-1)
497	Y2 = YSTP(J)
498	NC = 500
499	DO 200 IC=1,NC
500	YS = 0.5*(Y2+Y1)
501	CALL POINT(NP,Y,F,INDEX,FS,YS,INSLCT)
502	IF(III.EQ.3) INSLCT = 3
503	IS=INDEX(1)
504	IE=INDEX(INSLCT)
505	CALL AREA(NP,Y,F,YS,FS,IS,IE,IFLATO)
506	IF(IFLATO.EQ.0) RETURN
507	IF(IFLATO+IFLAT1 .LT. 0) THEN
508	Y2 = YS
509	IFLAT2 = IFLATO
510	ELSE
511	Y1 = YS
512	IFLAT1 = IFLATO
513	ENDIF
514	200 CONTINUE
515	WRITE(*,*) 'After ',NC,' steps of bisection'
516	RETURN
517	ELSE
518	IFLAT1 = IFLAT2
519	GOTO 500
520	ENDIF
521	C
522	FIRST = 1.
523	500 CONTINUE
524	RETURN
525	END

UNIX™  
FORTRAN ProgramSOURCE PROGRAM  
**LHF.f**DATE 7/14/94 PAGE #  
TIME 5:00:11 pm 9

LINE #

SOURCE TEXT

```
529 SUBROUTINE POINT(NP,Y,F,INDEX,FS,YS,INRCT)
530 DIMENSION Y(NP),F(NP)
531 DIMENSION INDEX(40),FS(40)
532
533 C Find the points FS on the F-function when YS is given
534 C INDEX = the index runs from 1 to NP
535 C INRCT = # of points being intersect, at least 3 point to do integration
536
537 INRCT = 0
538 IF(YS .LT. Y(1)) THEN
539   INRCT = INRCT + 1
540   FS(INRCT) = 0.
541   INDEX(INRCT) = 1
542 ENDIF
543
544 DO 100 I=2,NP
545   FAC1 = YS - Y(I)
546   FAC2 = YS - Y(I-1)
547   IF(FAC1*FAC2 .LE. 0.) THEN
548     IF(ABS(Y(I)-Y(I-1)) .LE. 1.E-14) THEN
549       write(*,*) 'ZEROOOOO',YS,Y(I),I
550       write(*,*) 'ZEROOOOO',YS,Y(I),I
551     INRCT = 0
552     RETURN
553   ENDIF
554   INRCT = INRCT + 1
555   SL = (F(I)-F(I-1))/(Y(I)-Y(I-1))
556   FS(INRCT) = F(I-1)+SL*(YS-Y(I-1))
557   INDEX(INRCT) = I
558
559 ENDIF
560 100 CONTINUE
561
562 IF(YS .GT. Y(NP)) THEN
563   INRCT = INRCT + 1
564   FS(INRCT) = F(NP)
565   Y(NP) = YS
566   INDEX(INRCT) = NP
567
568 RETURN
569 END
```

UNIX™  
FORTRAN ProgramSOURCE PROGRAM  
LHF.fDATE 7/14/94  
TIME 5:00:11 pm

PAGE # 10

## SOURCE TEXT

LINE # SUBROUTINE AREA(NP,Y,F,YS,FS,IS,IE,IFLAT)

571 DIMENSION Y(NP),F(NP),FS(40)

572 COMMON/SHOCK/ INSCT

573 C Fine the integral of F by trapezoidal rule

574 C Integrating from I=IS to IE, E1 is area that from YS to Y(IS)

575 C and E2 is the area that from Y(IE) to YS. Thus E2 should be

576 C subtracted out and E1 should be added in

577 C

578 C

579 C E1 = 0.5\*(Y(IS)-YS)\*(F(IS)+FS(1))

580 C IF(FS(1) .EQ. 0.) E1 = 0.

581 C E2 = 0.5\*(Y(IE)-YS)\*(F(IE)+FS(INSCT))

582 C AREA1 = E1

583 C IE=IE-1

584 C DO 10 I = IS,IE

585 C SLAP = 0.5\*(Y(I+1)-Y(I))\*(F(I+1)+F(I))

586 C AREA1 = AREA1 + SLAP

587 C

588 10 CONTINUE

589 C

590 C A = AREA1 - E2

591 C IF(A.GT.0) IFLAT=1

592 C IF(A.LT.0) IFLAT=-1

593 C IF(ABS(A).LT.1.E-7) IFLAT=0

594 C

595 C RETURN

596 C END

LINE #

SOURCE TEXT

```
598      SUBROUTINE WBODY(JDIM,S,TAU)
599      C This subroutine find the area distribution of
600      C the wing-body configuration.
601      DIMENSION S(JDIM),TAU(JDIM)
602      C
603      PI=4.*ATAN(1.)
604      ANG=21.*PI/180.
605      ANG1=80.*PI/180.
606      DX=25.52/FLOAT(JDIM-1)
607      C
608      S(1) = 0.
609      TAU(1) = 0.
610      DO 2 J=2,JDIM
611        TAU(J) = TAU(J-1)+DX
612        TTT = TAU(J)-7.01
613        IF(TTT.GT.0.) TTT=0.
614        RR=0.54-0.011*TTT**2
615        S(J) = PI*RR*RR
616        IF(TAU(J).GT.8.21 .AND. TAU(J).LT.12.25) THEN
617          AA = 4.*0.5*0.05*TAN(ANG)*(TAU(J)-8.21)**2
618          S(J) = S(J) + AA
619        ENDIF
620        IF(TAU(J).GT.12.25 .AND. TAU(J).LT.15.77688849) THEN
621          B2 = 0.05*(16.29-TAU(J))
622          B2 = 2.91*((TAU(J)-12.25)/(15.77688849-12.25))
623          B1 = (TAU(J)-8.21)*TAN(ANG)-B2
624          B1 = 0.05*B1/TAN(ANG)
625          AA = 4.*(0.5*B1*B1+0.5*(B1+B2)*B2)
626          S(J) = S(J) + AA
627        ENDIF
628        IF(TAU(J).GT.15.77688849 .AND. TAU(J).LT.16.29) THEN
629          AA = 4.*(0.5*0.05*TAN(ANG))*(16.29-TAU(J))**2
630          S(J) = S(J) + AA
631        ENDIF
632        IF(TAU(J).GT.18.93 .AND. TAU(J).LT.17.52) THEN
633          SLOP=(0.15-0.54)/(17.93-17.52)
634          RRR=0.54+SLOP*(TAU(J)-17.52)
635          S(J) = PI*RRR**2
636        ENDIF
637        IF(TAU(J).GT.17.93) S(J)=PI*0.15*0.15
638
2      CONTINUE
639      RETURN
640
641
```

LINE #

SOURCE TEXT

```
643 SUBROUTINE CONE(JDIM,S,TAU)
644 C This subroutine find the area distribution of the cone-cylinder
645 C with half-angle 3.24 degree and 8.6 units of length.
646 C
647 C
648 PI=4.*ATAN(1.)
649 ANG = 3.24*PI/180.
650 DX = 16./FLOAT(JDIM-1)
651 S(1) = 0.
652 TAU(1) = 0.
653 DO 2 J=2,JDIM
654   TAU(J)=TAU(J-1)+DX
655   IF(TAU(J).LE.8.6) THEN
656     R = TAU(J)*TAN(ANG)
657   ELSE
658     R = 8.6*TAN(ANG)
659   ENDIF
660   S(J) = PI*R*R
661 CONTINUE
662 RETURN
663 END
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM  
LHF.f

DATE	7/14/94	PAGE #
TIME	5:00:11 pm	13

LINE #

	SOURCE TEXT
665	SUBROUTINE SEARS(JDIM,S,TAU)
666	C This subroutine find the area distribution of the Sears-Haack body
667	C with fineness ratio 23.5.
668	C DIMENSION S(JDIM),TAU(JDIM)
669	C
670	C # of point on body + # of point on sting = JDIM
671	JBODY = JDIM*(2./3.)
672	JSNG = JDIM - JBODY
673	C
674	PI=4.*ATAN(1.)
675	BL = 1.
676	TOTL = 1.9* BL
677	F = 23.5
678	DTHETA = PI/FLOAT(JBODY-1)
679	DX = BL/FLOAT(JBODY-1)
680	RMAX = BL/(2.*F)
681	S(1) = 0.
682	TAU(1) = 0.
683	C
684	Constants of Sears-Adams body
685	write(*,*)'Input Abase/Amax'
686	read(*,*) AR
687	write(*,*)'Input XMAX'
688	read(*,*) AA
689	XMAX = AA*BL
690	CONST = AR/PI
691	C1 = 1./(2.* (2.*XMAX/BL - 1.))
692	C
693	DO 2 J=2,JBODY
694	C THETA = PI-DTHETA*FLOAT(J-1)
695	C TAU(J) = (1.+COS(THETA))*BL/2.
696	C TAU(J) = FLOAT(J-1)*DX
697	C THETA = ACOS(2.*TAU(J)/BL - 1.)
698	C Sears-Haack body
699	C POS = (SIN(THETA))**3
700	C Haack-Adams body
701	POS = CONST*( PI-THETA+0.5*SIN(2.*THETA) +
702	(4./3.)*C1*(SIN(THETA))**3 )
703	C
704	R = RMAX*SQRT( ABS(POS) )
705	S(J) = PI*R*R
706	2 CONTINUE
707	C
708	Add a sting
709	DX = (TOTL-BL)/FLOAT(JSNG)
710	DO 5 J=JBODY+1,JDIM
711	TAU(J) = TAU(J-1) + DX
712	S(J) = S(J-1)
713	5 CONTINUE
714	RETURN
715	END

## SOURCE TEXT

LINE #	SOURCE TEXT
716	SUBROUTINE BULLET(JDIM,S,TAU)
717	c This subroutine find the area distribution of a bullet with a form
718	R = AX*gama
719	DIMENSION S(JDIM),TAU(JDIM)
720	c
721	PI=4.*ATAN(1.)
722	BL = 4.
723	GAMA = 0.65
724	RBASE = 0.25
725	A = RBASE/(BL**GAMA)
726	TOTLEN = BL + 2.*BL
727	DX = TOTLEN/FLOAT(JDIM-1)
728	S(1) = 0.
729	TAU(1) = 0.
730	DO 2 J=2,JDIM
731	TAU(J)=TAU(J-1)+DX
732	IF(TAU(J).GE.BL) THEN
733	R = RBASE
734	ELSE
735	R = A*TAU(J)**GAMA
736	ENDIF
737	S(J) = PI*R*R
738	CONTINUE
739	RETURN
740	END

LINE #

SOURCE TEXT

```
742 SUBROUTINE WING(JDIM,S,TAU)
743 C This subroutine find the area distribution of the low-aspect-ratio wing
744 DIMENSION S(JDIM),TAU(JDIM)
745 C
746 PI=4.*ATAN(1.)
747 STING = 0
748 DX = 3./FLOAT(JDIM-1)
749 S(1) = 0.
750 TAU(1) = 0.
751 DO 2 J=2,JDIM
752     TAU(J)=TAU(J-1)+DX
753     IF(TAU(J).LT.2.) THEN
754         Z = (PI/12.5)*(TAU(J)-0.5*TAU(J)*TAU(J))
755         S(J) = Z
756         IF(TAU(J).GT.1.70897) THEN
757             STING=PI*0.0625*0.0625
758             S(J) = Z + STING - Z*0.125
759         ENDIF
760     ELSE
761         STING=PI*0.0625*0.0625
762         S(J) = STING
763     ENDIF
764 2 CONTINUE
765 RETURN
766 END
```

LINE #

SOURCE TEXT

```
767 SUBROUTINE BFUNC(JDIM,S,TAU)
768 C This subroutine obtains the B-function from fort.10 and add it
769 C into the equivalent area.
770 C PARAMETER (NMAX=800)
771 C DIMENSION S(JDIM),TAU(JDIM),B(NMAX),X(NMAX)
772 C COMMON/PAR/ FMACH,PFAC
773 C
774 C OPEN(UNIT=33,FILE='coef.dat',FORM='FORMATTED')
775 C
776 READ(33,12)
777 READ(33,12)
778 READ(33,12)
779 READ(33,12)
780 READ(33,12)
781 DO 10 I=1,NMAX
782 C     READ(33,15,END=17) X(I),CL,CD,SLOD,B(I),CM
783 C     READ(33,* ,END=17) X(I),B(I)
784 B(I) = B(I)*PFAC
785 10 CONTINUE
786 11 CONTINUE
787 12 FORMAT(1X)
788 15 FORMAT(6E13.5)
789 17 CONTINUE
790 CLOSE(33)
791 NPOINT = I-1
792 OPEN(UNIT=33, FILE='bfn.dat')
793 DO 20 I=1,NPOINT
794 WRITE(33,*) X(I),B(I)
795 20 CONTINUE
796 C
797 ISTART=1
798 DO 50 J=1,JDIM
799 DO 30 I=ISTART,NPOINT
800 IF(ABS(X(I)-TAU(J)) .LE. 1.E-10) THEN
801 S(J)=S(J)+B(I)
802 ISTART=I
803 GOTO 40
804 ENDIF
805 IF(X(I) .GT. TAU(J)) THEN
806 IF(I.EQ.1) THEN
807 BF=0.
808 IF=0.
809 ELSE
810 BF=B(I-1)
811 XF=X(I-1)
812 ENDIF
813 SLOPE=(B(I)-BF)/(X(I)-XF)
814 BT = B(I) + SLOPE*(TAU(J)-X(I))
815 S(J) = S(J) + BT
816 ISTART=I-1
817 IF(I.EQ.1) ISTART=1
818 GOTO 40
819 ELSE
820 IF(I .LT. NPOINT) GOTO 30
821 S(J) = S(J) + B(NPOINT)
822 ISTART=I
823 GOTO 40
824 ENDIF
825 30 CONTINUE
826 40 CONTINUE
827 50 CONTINUE
828 RETURN
829 END
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM

LHF.f

DATE 7/14/94

PAGE #

TIME 5:00:11 pm

17

LINE #

SOURCE TEXT

```
830      SUBROUTINE WB(JDIM,SP,TAU)
831      C   This subroutine obtains the wing-body interference correction
832      C   and add it into the derivative of equivalent area.
833      C   This is a test for wing-body case
834      DIMENSION SP(JDIM),TAU(JDIM)
835      COMMON/PAR/ FMACH,PFAC
836      C
837      DO 10 J=1,JDIM
838         IF(TAU(J).GE.8.21 .AND. TAU(J).LE.12.25)
839         SP(J)=SP(J)+4.*.05*.54
840         IF(TAU(J).GT.12.25 .AND. TAU(J).LE.16.29)
841         SP(J)=SP(J)-4.*.05*.54
842 10    CONTINUE
843      RETURN
844      END
```

LINE #	SOURCE TEXT
846	SUBROUTINE FUNC(TAU,FTAU,JDIM)
847	DIMENSION TAU(JDIM),FTAU(JDIM)
848	NAMELIST /FFUNC/ YF,ELAM,C,B,B,D,E,BL,YR,DEL
849 C	Read the input parameters
850	READ(3,FFUNC)
851	WRITE(6,FFUNC)
852	
853	TAU(1)=0.
854	FTAU(1)=0.
855	DY-YR/FLOAT(JDIM-1)
856	DO 10 J=2,JDIM
857	TAU(J)=TAU(J-1)+DY
858	IF(YF.EQ.0.) GOTO 6
859	IF(TAU(J).LE.YF/2.) FTAU(J)=2.*TAU(J)*B/YF
860	IF(TAU(J).GE.YF/2.0 .AND. TAU(J).LE.YF)
861	FTAU(J)=C*(2.*TAU(J)/YF-1.) - B*(2.*TAU(J)/YF-2.)
862	
863 C	IF(TAU(J).GE.YF .AND. TAU(J).LE.ELAM)
864	FTAU(J)=B*(TAU(J)-YF)+C
865	IF(TAU(J).GE.YF .AND. TAU(J).LE.DEL) FTAU(J)=C
866	IF(TAU(J).GE.DEL .AND. TAU(J).LE.ELAM)
867	FTAU(J)=B*(TAU(J)-DEL)+C
868	
869 C	IF(TAU(J).GE.ELAM .AND. TAU(J).LE.BL)
870	FTAU(J)=B*(TAU(J)-ELAM)+(ELAM*B-D)
871	IF(TAU(J).GE.BL)
872	FTAU(J)=(ELAM*B-D+B*(BL-ELAM))+(TAU(J)-YR)/(BL-YR)
873	FTAU(J)=E/(TAU(J)-(BL-ABS(BL-ELAM)/10.))
874	
875	10 CONTINUE
876 C	CALL DISTARC(TAU,FTAU,JDIM,TAU,FTAU,JDIM,10.,0)
877	
878	WRITE(13,75)
879	DO 20 J=1,JDIM
880	WRITE(13,80) TAU(J),FTAU(J)
881	
882	20 CONTINUE
883	75 FORMAT(22E16.8-function from input)
884	80 FORMAT(2X,F8.4,1X,E16.8)
885	RETURN
886	END

LINE #

SOURCE TEXT

```
888 SUBROUTINE EAREA(S,FTAU,TAU,JDIM)
889 DIMENSION S(JDIM),TAU(JDIM),FTAU(JDIM)
890 DIMENSION F(900)
891 C
892 C Obtain the equivalent area from F-function via Abel Transform
893 C
894 C A(x) =  $\int_{-\infty}^x \int_0^\infty F(t)/\sqrt{t-y} dt dy$ 
895 C
896 C
897 C
898 S(1) = 0.
899 TAU(1)=0.
900 DO 10 J=2,JDIM
901 SS = 0.
902 DO 7 I=1,J-1
903 DY=TAU(I+1)-TAU(I)
904 FINGRL = 0.
905 DO 5 K=1,I-1
906 DT = TAU(K+1)-TAU(K)
907 FINGRL = FINGRL + DT*FTAU(K)/SQRT(TAU(I)-TAU(K))
908 5 CONTINUE
909 SS = SS + 2.* FINGRL*DY
910 C SS = SS + 4.*SQRT(TAU(J)-TAU(I))*FTAU(I)*DY
911 7 CONTINUE
912 S(J) = SS
913 10 CONTINUE
914 TANN=S(3)/TAU(3)
915 S(2)=TANN*TAU(2)
916 WRITE(12,15)
917 15 FORMAT(27HArea from given F-function)
918 DO 20 J=1,JDIM
919 WRITE(12,80) TAU(J),S(J)
920 20 CONTINUE
921 80 FORMAT(2X,F8.4,1X,E16.8)
922 RETURN
923 END
```

LINE #	SOURCE TEXT
925	SUBROUTINE RISETIME(FMACH,P,T,ALT,IRISE)
926	C An empirical method to calculate the rise time of a sonic boom
927	C Rise time derived from regression analysis of Air Force sonic boom
928	C flight test data. Good for N-wave type of signal, may be somewhat
929	C conservative (shorter rise time).
930	C All unit used are English unit !!!
931	C
932	C FMACH = Free-stream Mach number
933	C P(T) = sonic boom
934	C PSH = Shock strength.
935	C ALT = Altitude (ft)
936	C P0 = Free-stream pressure (lb/ft <sup>2</sup> )
937	C RT = Rise time (sec)
938	C TEMP = Temperature R=F+459.67-(9/5)K
939	C DIMENSION P(NP),T(NP)
940	C P0 = 2116.2
941	C TEMP = 518.69
942	C
943	I COUNT = 0
944	12 CONTINUE
945	I COUNT = I COUNT + 1
946	C Find out the shock strength
947	C PSH = 0.
948	C ISHO = 0.
949	DO 30 I=1,NP
950	IF(T(I).EQ.T(I+1)) THEN
951	IF(ISHO.EQ.0) ISHO=I
952	PSH = ABS(P(I+1)-P(ISHO))
953	ELSE
954	IF(PSH.EQ.0.) THEN
955	GOTO 30
956	ELSE
957	GOTO 40
958	ENDIF
959	ENDIF
960	30 CONTINUE
961	40 CONTINUE
962	ISH = I
963	IF(PSH.EQ.0.) RETURN
964	C
965	IF(IRISE.EQ.1) THEN
966	Now calculate the rise time using Air Force data base
967	Y1 = 2.92*FMACH - 7.38
968	Y2 = Y1 + ((8.5*FMACH**2 - 45.9*FMACH + 62.9)**(.5))
969	AK1 = Y2 * 1000.
970	AK = AK1 / (((ALT/1000.) - 5.)*2117.)
971	VIS = 100. + 0.5*(TEMP-410.)
972	RT = (AK*VIS)*P0/(PSH*TEMP)
973	RT = RT/1000.
974	ELSEIF(IRISE.EQ.2) THEN
975	Now calculate the rise time assuming Iperf has same rise time
976	RT = 0.003/PSH
977	ENDIF
978	C
979	WRITE(14,80)RT
980	80 FORMAT(3B8) Rise time (sec) of the bow shock is, F10.5)
981	C Originally, T(ISHO)-T(ISH) with infinite shock strength, now create a
982	C signal with the rise time, between the index ISHO to ISH
983	C Also extend the signal by the amount of rise time.
984	C
985	DRT = RT/FLOAT(ISH-ISHO)
986	DO 200 I=ISHO,ISH-1
987	T(I+1) = T(I) + DRT
988	200 CONTINUE
989	DO 300 I=ISH+1,NP
990	T(I) = T(I) + RT
991	300 CONTINUE
992	C
993	IF(ICOUNT.LT.10) GOTO 12
994	RETURN
995	END
996	C
997	C

LINE #

SOURCE TEXT

```
999 ****
1000 SUBROUTINE DISTARC(X,Y,N,XNEW,YNEW,NNEW,FAC,IFLAT)
1001 C
1002 C DIMENSION X(N),Y(N),XNEW(NNEW),YNEW(NNEW)
1003 C
1004 C This program redistribute the points (X,Y) by subroutine DISTRI
1005 C based on the arc length. FAC is the first grid spacing. Note that
1006 C the end points of the two sets are the same.
1007 C IFLAT=0, grid points will cluster near the first point, -1 near the end.
1008 C Input array is (X(i),Y(i)), i=1,...,N
1009 C Output array is (XNEW(i),YNEW(i)), i=1,...,NNEW
1010 C
1011 C
1012 C PARAMETER (MAX=2000)
1013 C DIMENSION S(MAX),TOTARC(MAX),XN(MAX),YN(MAX)
1014 C
1015 C Maximum number of points allowed is MAX
1016 C IF(MAX.LE.N .OR. MAX.LE.NNEW) THEN
1017 C   WRITE(*,*) 'SUB DISTARC : MAX is less than N or NNEW'
1018 C   STOP
1019 C ENDIF
1020 C
1021 C Look for total arc length
1022 C TOTARC(1) = 0.
1023 C DO 10 K=2,N
1024 C   ARC = SQRT( (X(K)-X(K-1))**2 + (Y(K)-Y(K-1))**2 )
1025 C   TOTARC(K) = TOTARC(K-1) + ARC
1026 C 10 CONTINUE
1027 C
1028 C Apply subroutine DISTRI to obtain the stretching function S
1029 C IF(FAC.LT.1.) THEN
1030 C   DELT=FAC*(TOTARC(N)/FLOAT(NNEW-1))
1031 C   CALL DISTRI(DELT,NNEW,S,IFLAT)
1032 C ELSE
1033 C   S(1) = 0.
1034 C   DO 25 K=2,NNEW
1035 C     S(K) = S(K-1) + 1./FLOAT(NNEW-1)
1036 C 25 CONTINUE
1037 C ENDIF
1038 C
1039 C Redistribution, put new array in temporary arrays XN and YN
1040 C XN(1)=X(1)
1041 C YN(1)=Y(1)
1042 C XN(NNEW)=X(N)
1043 C YN(NNEW)=Y(N)
1044 C
1045 C DO 60 J = 2,NNEW
1046 C   ARCREW = S(J)*TOTARC(N)
1047 C   DO 55 K = 2,N
1048 C     IF(TOTARC(K).EQ.ARCREW) THEN
1049 C       XN(J) = X(K)
1050 C       YN(J) = Y(K)
1051 C       GOTO 60
1052 C     ENDIF
1053 C     IF(TOTARC(K).GT.ARCREW) THEN
1054 C       X1 = X(K-1)
1055 C       X2 = X(K)
1056 C       Y1 = Y(K-1)
1057 C       Y2 = Y(K)
1058 C       XX = X1 + (X(K)-X(K-1))*
1059 C            (ARCREW-TOTARC(K-1))/(TOTARC(K)-TOTARC(K-1))
1060 C       CALL LININT(X1,X2,Y1,Y2,XX,YY)
1061 C       XN(J) = XX
1062 C       YN(J) = YY
1063 C     GOTO 60
1064 C   ENDIF
1065 C 55 CONTINUE
1066 C 60 CONTINUE
1067 C
1068 C
1069 C Write the temporary arrays into the output XNEW, YNEW
1070 C DO 70 J=1,NNEW
1071 C   XNEW(J) = XN(J)
1072 C   YNEW(J) = YN(J)
1073 C
1074 C 70 CONTINUE
1075 C RETURN
1076 C END
```

## SOURCE TEXT

LINE #	COLUMN 1
1077	*****
1078	SUBROUTINE DISTRI(FANG,KFCS,S,IFINE)
1079	PARAMETER (MAX=500)
1080	DIMENSION S(MAX),DUM(MAX)
1081	C..... Calculating the stretching function S when given
1082	the first spacing, FANG, and the number of points KFCS
1083	C..... if IFINE=1, distribution is clustering at outer grid
1084	C..... if IFINE>1, distribution is uniform
1085	C.....
1086	IF(MAX.LE.KFCS) THEN
1087	WRITE(*,*)'SUB DISTRI : MAX is less than KFCS'
1088	STOP
1089	ENDIF
1090	IF(KFCS.EQ.1) THEN
1091	S(1) = 0.
1092	GOTO 40
1093	ENDIF
1094	C.....
1095	DZ1 = FANG
1096	KFM = KFCS-1
1097	DZETA = 1./FLOAT(KFM)
1098	RDBETA = 1.5
1099	CALL GRBET(DZ1,KFM,0.0001,100,RDBETA)
1100	CALL F21(KFCS,RDBETA,DZETA,S)
1101	C.....
1102	IF (IFINE.EQ.1) THEN
1103	DO 37 K=1,KFCS
1104	DUM(KFCS-K+1) = S(K)
1105	CONTINUE
1106	37
1107	DO 38 K=1,KFCS
1108	S(K) = 1.-DUM(K)
1109	CONTINUE
1110	38
1111	ENDIF
1112	CONTINUE
1113	RETURN
	END

LINE #

LINE #	SOURCE TEXT
1114	*****
1115	SUBROUTINE F21(L1,TBETA,DET,2)
1116	C
1117	COMPUTES NORMALIZED NORMAL DISTANCE, Z(L)
1118	C
1119	DIMENSION Z(250)
1120	IF(TBETA.EQ.1.) THEN
1121	DO 10 L=1,L1
1122	Z(L)=0.
1123	10 CONTINUE
1124	ELSE
1125	DO 20 L=1,L1
1126	ETA=(L-1)*DET
1127	RR=(TBETA+1.)/(TBETA-1.)
1128	EEE=1.-ETA
1129	RBETA=RR**EEE
1130	Z(L)=(TBETA-1.)*(RR-RBETA)/(RBETA+1.)
1131	20 CONTINUE
1132	END IF
1133	RETURN
1134	END

UNIX™  
FORTRAN Program

SOURCE PROGRAM

LHF.f

DATE 7/14/94  
TIME 5:00:11 pm

24

SOURCE TEXT

```
LINE # SUBROUTINE GRBET(DFM,NPT,FPCC,ICC,BETA)
1135 C
1136 C BISECTION METHOD USED TO DETERMINE STRETCHING PARAMETER, BETA,
1137 C WHICH GIVES DESIRED CT AT THE WALL
1138 C
1139 C
1140 DIMENSION Z(250)
1141 ICC=ICC
1142 FPCC=FPCC*DFM
1143 BETA1=BETA
1144 Z1=DFM
1145 DET=1./NPT
1146 BR=1.
1147 FR=21
1148 IICC=ICC/10
1149 DO 10 I=1,IICC
1150 BF=BETA1
1151 BETA=0.5*(BETA1+1.)
1152 CALL FZ1(2,BF,DET,Z)
1153 FF=Z(2)-Z1
1154 IF(FF.GT.0.) GO TO 15
1155 BETA=2.*BETA1-1.
1156 10 CONTINUE
1157 15 CONTINUE
1158 DO 5 NIT=1,ICC
1159 CALL FZ1(2,BETA,DET,Z)
1160 F=Z(2)-Z1
1161 IF(F.GT.0.) THEN
1162 FF=F
1163 BF=BETA
1164 ELSE
1165 FR=F
1166 BR=BETA
1167 END IF
1168 BETA=0.5*(BF+BR)
1169 IF(ABS(F).LT.FPCC) GO TO 4
1170 5 CONTINUE
1171 WRITE(6,100) BETA,F
1172 100 FORMAT(1HO,36E EXCEEDED MAX. NO. OF ITS....BETA,F ,3G13.6)
1173 C
1174 C 4 CONTINUE
1175 C     CALL FZ1(2,BETA,DET,E)
1176 C     F=Z(2)-Z1
1177 C     BM1=BETA-1.
1178 C     BFM1=BF-1.
1179 C     BRM1=BR-1.
1180 C     RETURN
1181 C
1182 END
```

LINE #

SOURCE TEXT

```
1182      SUBROUTINE LININT(X1,X2,Y1,Y2,XLOCAL,YLOCAL)
1183      C This subroutine linearly interpolate YLOCAL when given (X1,Y1) & (X2,Y2)
1184      IF(X1.EQ.X2) THEN
1185          YLOCAL=(Y2-Y1)/2.
1186          GOTO 100
1187      ENDIF
1188      SLOPE = (Y2-Y1)/(X2-X1)
1189      YLOCAL = SLOPE*(XLOCAL-X2) + Y2
1190 100    CONTINUE
1191    RETURN
1192    END
```

LINE #	SOURCE TEXT
194	SUBROUTINE MARCH(NMAX,NP,Y,YSTP,F)
195	DIMENSION F(NP),Y(NP)
196	DIMENSION YSTP(NMAX)
197	DIMENSION INDEX(40),FS(40)
198	COMMON/SHOCK/ INSCT
199	c This subroutine marches the Y direction, and
200	c check if the areas are balanced and then place the shock
201	c
202	c
203	KOUNT = 0
204	YEND = Y(NP)
205	100 CONTINUE
206	DO IND=1,40
207	INDEX(IND)=0.
208	INSCT = 0
209	ENDDO
210	CALL SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,0)
211	c For tail shock, no need to check the possible position of shock
212	IF(YS.GT.0.7*(YSTP(NMAX)-YSTP(1))) GOTO 400
213	c Only one possible location of shock
214	IF(INSCT.EQ.3) GOTO 400
215	c More than one possible locations of shock
216	IF(INSCT.GE.5) THEN
217	YSHK = YS
218	CALL SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,3)
219	IF(YSHK.LT.YS) THEN
220	The wing shock overcomes
221	CALL SHKPT(NMAX,NP,Y,YSTP,F,INDEX,FS,YS,0)
222	GOTO 400
223	ELSE
224	There are two separated shocks
225	For the shock is actually locate on the turning edge of F-function
226	we need to relocate it
227	Fix the Y1 and Y2 of this small region
228	IF(YS.GT.Y(INDEX(INSCT)+1)) THEN
229	Y2 = YS
230	BIG = 0.
231	DO ITEST=INDEX(INSCT)+1,NP
232	IF(YS.GT.Y(ITEST) .AND. ABS(YS-Y(ITEST)).GT.BIG) THEN
233	Y1 = Y(ITEST)
234	BIG = ABS(YS-Y(ITEST))
235	ENDIF
236	IF(Y(ITEST).GE.YS) GOTO 300
237	ENDDO
238	300 CONTINUE
239	c Find YS by bisecting Y1 and Y2
240	NC = 500
241	DO 320 IC=1,NC
242	YS = 0.5*(Y2+Y1)
243	CALL POINT(NP,Y,F,INDEX,FS,YS,INSCT)
244	DO II=INSCT,1,-1
245	IF(INDEX(II).LE.ITEST) THEN
246	INSCT = II
247	GOTO 310
248	ENDIF
249	ENDIF
250	310 CONTINUE
251	IS=INDEX(1)
252	IE=INDEX(INSTC)
253	CALL AREA(NP,Y,F,YS,FS,IS,IE,IFLATO)
254	IF(IFLATO.EQ.0) GOTO 400
255	IF(IFLATO.GT.0) THEN
256	Y2 = YS
257	ELSE
258	Y1 = YS
259	ENDIF
260	320 CONTINUE
261	WRITE(*,*) 'After ',NC,' steps of bisection'
262	ENDIF
263	GOTO 400
264	ENDIF
265	ENDIF
266	400 CONTINUE
267	c Form the shock
268	c
269	IF(INSCT.LE.1 .AND. YS.GE.YEND) RETURN
270	FDIS = FS(INSCT)-FS(1)
271	if(FLOAT(INDEX(INSCT)-INDEX(1)).eq.0.0) then
272	WRITE(15,*) 'SDT: ZERO DIVISION ABOUT TO HAPPEN in MARCH'
273	df = 1.e32
274	else
275	df = FDIS/FLOAT(INDEX(INSCT)-INDEX(1))
276	endif
277	F(INDEX(1)) = FS(1)
278	Y(INDEX(1)) = YS
279	IS = INDEX(1) + 1
280	DO 450 I = IS,INDEX(INSCT)
281	Y(I) = YS
282	F(I) = F(I-1) + DF
283	450 CONTINUE
284	IF(KOUNT.EQ.20) THEN
285	WRITE(*,*) 'KOUNT=20!!'
286	RETURN
287	ELSE
288	KOUNT = KOUNT+1
289	GOTO 100
290	ENDIF
291	END

UNIX™  
FORTRAN Program

SOURCE PROGRAM

LHF.f

DATE

7/14/94

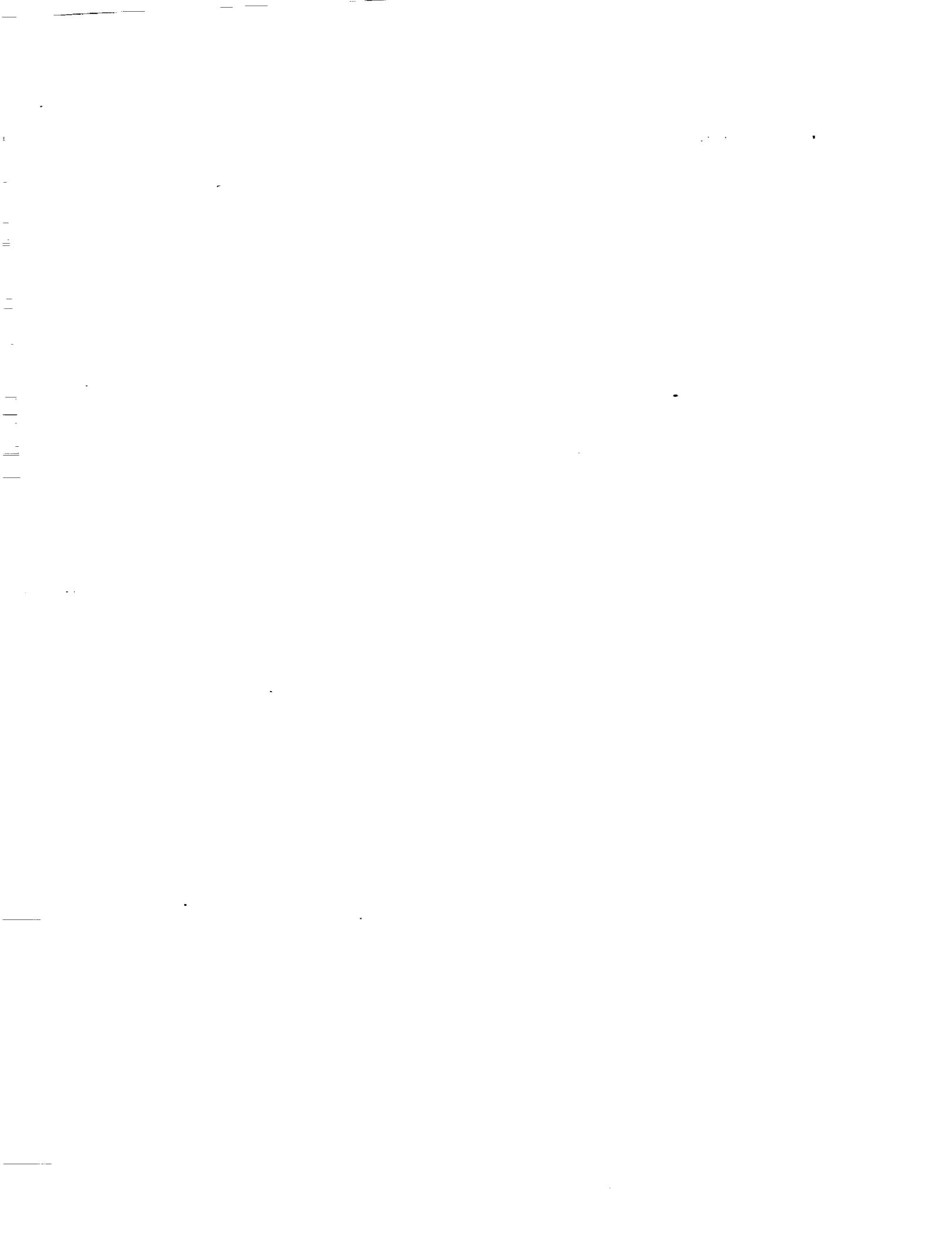
PAGE #

TIME

5:00:11 pm

27

LINE #	SOURCE TEXT
1301	SUBROUTINE SIMPSON(X,F,M,X0,X1,SUM)
1302	DIMENSION X(M),F(M)
1303	C
1304	C    M is odd
1305	N = M/2
1306	SUM = 0.
1307	DO 10 I=1,N
1308	ODD = ODD + 2.*F(2*I+1)
1309	EVEN= EVEN + 4.*F(2*I)
1310	CONTINUE
1311	C
1312	SUN = F(1) + ODD + EVEN + F(M)
1313	SUM = SUM*(X1-X0)/(6.*FLOAT(N))
1314	RETURN
1315	
1316	END



# Appendix B

## SAMGRID (Fortran Listing)



# UNIX™ FORTRAN Program

SOURCE PROGRAM  
**samgrid.f**

DATE	7/07/94	PAGE #
TIME	4:18:56 pm	1

LINE #	SOURCE TEXT
1	PROGRAM SAMGRID
2	include "sgrid.com"
3	c Dr. Samson Cheung
4	c Date: Dec., 1993
5	c This subroutine reads a surface grid in airfoils
6	c sections and reformats it to produce
7	c a surface grid of axisymmetric cross-sections.
8	c
9	c Data: Dec., 1993 Version 1.0
10	c
11	c read input geometry
12	c
13	c
14	c OPEN(UNIT=10,FILE='NBSGRID.IN',STATUS='OLD',FORM='FORMATTED')
15	c OPEN(UNIT=30,FILE='NACGRID.IN',STATUS='OLD',FORM='FORMATTED')
16	c OPEN(UNIT=40,FILE='XZSIZE.IN',STATUS='OLD',FORM='FORMATTED')
17	c
18	c
19	c
20	c c NSEC = # of sections (streamwise stations) of the new grid
21	c NPTS = # of pts in the circumferential direction (MUST be odd)
22	c NY=MAX streamwise stations
23	c FAC = the first grid spacing in DISTRI
24	c XLE = X leading edge
25	c ARNING = 0, arrow wing
26	c KTIP = number of points in the wrap-around direction on one surface
27	c NC=number of cut in the spanwise direction
28	c ND=number of points in the upper part of the wing
29	c NL=number of points in the lower part of the wing
30	c
31	c
32	c Read surfgrid dimensions (sssec x npts x 1)
33	c NAMELIST/WING/ NSEC,NPTS,FAC
34	c READ(40,WING)
35	c NNRITE(*,WING)
36	c
37	c Read the input grid
38	c
39	c CALL WINGIN
40	c
41	c
42	c Setup distribution of cross-sections to be obtained (xdist)
43	c XRL-E-XLE(1)
44	c XRT-E-XLE(1)+CHORD(1)
45	c WRTE-XLE(NC)+CHORD(NC)
46	c IF(XRTE.GE.WRTE) THEN
47	c    XRT = XRTE
48	c    ARNING = -1.
49	c ELSE
50	c    XRT = WRTE
51	c    ARNING = 1.
52	c ENDIF
53	c WAKE = AMIN1(XRTE,WRTE)
54	c DO 100 J=1,NSEC
55	c    XDIST(J)=XRL-E+(XRT-XRLE)*(FLOAT(J-1)/FLOAT(NSEC-1))
56	c CONTINUE
57	c
58	c KTIP=(NPTS+1)/2
59	c
60	c The nose of the wing
61	c DO 187 K=1,NPTS
62	c    XOUT(1,K)=XDIST(1)
63	c    YOUT(1,K)=YBASE(1,1,1)
64	c    ZOUT(1,K)=ZBASE(1,1,1)
65	c
66	c 187 CONTINUE
67	c
68	c Begin main loop for each x-station
69	c
70	c DO 1000 L=2,NSEC
71	c
72	c    XLOCAL-XDIST(L)
73	c
74	c Redistribute the points from spanwise cut to streamwise cut.
75	c The output ZDIST and YNEW are from the root to the tip, therefore
76	c when doing the lower surface, need to rearrange the argument.
77	c The output (ZDIST,YNEW) in both surfaces have KTIP / of pts in
78	c the circumferential direction, their last point have the same physical
79	c value for both surfaces.
80	c
81	c
82	c Do the lower surface
83	c CALL REDIST(XLOCAL,KTIP,FAC,1)
84	c DO 300 K=1,KTIP
85	c    XOUT(L,K) = XLOCAL
86	c    YOUT(L,K) = YNEW(K)
87	c    ZOUT(L,K) = ZDIST(K)
88	c 300 CONTINUE
89	c
90	c Do the upper surface
91	c CALL REDIST(XLOCAL,KTIP,FAC,2)
92	c DO 400 K=KTIP+1,NPTS
93	c    XOUT(L,K) = XLOCAL
94	c    YOUT(L,K) = YNEW(NPTS-K+1)
95	c    ZOUT(L,K) = ZDIST(NPTS-K+1)
96	c 400 CONTINUE
97	c
98	c For the computational grid of UP3D code
99	c the wake has to have two different pts in same
100	c physical location, such that (Y1,Z1)=(Y1,Z2). Here
101	c the calculation divided into upper and lower parts,
102	c for safety sake, set Z1-Z2
103	c IF(XLOCAL.LT.WAKE) GOTO 900
104	c DO 500 K=1,KTIP-1
105	c    K1=KTIP+K
106	c    K2=KTIP-K
107	c *Hagland model    IF(ABS(YOUT(L,K1)-YOUT(L,K2)).LE. 3.0E-4) THEN
108	c       IF(ABS(YOUT(L,K1)-YOUT(L,K2)).LE. 1.0E-2) THEN
109	c         IF(ABS(YOUT(L,K1)-YOUT(L,K2)).LE. 1.0E-5) THEN
110	c           ZOUT(L,K1)=ZOUT(L,K2)
111	c           YOUT(L,K1)=YOUT(L,K2)
112	c       ENDIF
113	c 500 CONTINUE
114	c 900 CONTINUE
115	c
116	c Proceed to next x coordinate
117	c
118	c 1000 CONTINUE
119	c
120	c Write out new surfgrid in plot3d format

LINE #

SOURCE TEXT

```
121 C
122   KN=1
123   WRITE(50)NPTS,KM,NSEC
124   DO 1234 L=1,NSEC
125     WRITE(50)(IOUT(L,K),K=1,NPTS),
126     (IOUT(L,K),K=1,NPTS),
127     (ZOUT(L,K),K=1,NPTS)
128 1234 CONTINUE
129 C   Write out original database in plotid format
130 C
131 C
132   N1=NU+NL
133   WRITE(11)N1,NC,KM
134   WRITE(11)((XBASE(I,1,M),I=1,NU),(XBASE(I,2,M),I=NL,1,-1),
135     M=1,NC),
136     ((YBASE(I,1,M),I=1,NU),(YBASE(I,2,M),I=NL,1,-1),
137     M=1,NC),
138     ((ZBASE(I,1,M),I=1,NU),(ZBASE(I,2,M),I=NL,1,-1),
139     M=1,NC)
140 C   Read the fuselage grid and combine the fuselage with the
141 C   wing grid to form a whole configuration.
142 C
143   CALL WBGRID
144 C
145 C   Read the nacelles grid and combine the nacelles with the
146 C   wing-body grid.
147 C
148   CALL NACGRID
149 C
150 C   CLOSE(10)
151 C   CLOSE(30)
152 C   CLOSE(40)
153 C   STOP
154 C   END
```

LINE #

```
*****  
154  
155      SUBROUTINE ADDGRID(NPL1,NPL2,X,Y,Z,NPI,NSEC,KDIM)  
156  
157      C This subroutine allows us to add a grid line between streamwise section  
158      C NPL1 and NPL2, and the new dimension is NSEC again  
159      C  
160      PARAMETER (MAX=400)  
161      DIMENSION YTEMP(MAX),ZTEMP(MAX)  
162      DIMENSION X(NPI),Y(NPI,NPI),Z(NPI,NPI)  
163  
164      IF(MAX.LE.NPI) THEN  
165      WRITE(*,*)'SUB ADDGRID : MAX is less than NPI'  
166      STOP  
167      ENDIF  
168      IF(NPL1.GE.NPL2) THEN  
169      WRITE(*,*)'No plane is added in the streamwise direction'  
170      STOP  
171      ENDIF  
172  
173      C Interpolating the new grid, and put it in a temporary array  
174      X1 = X(NPL1)  
175      X2 = X(NPL2)  
176      XX = 0.5*(X(NPL1)+X(NPL2))  
177      DO 10 K=1,KDIM  
178      Y1 = Y(NPL1,K)  
179      Y2 = Y(NPL2,K)  
180      Z1 = Z(NPL1,K)  
181      Z2 = Z(NPL2,K)  
182      CALL LININT(X1,X2,Y1,Y2,XX,YY)  
183      CALL LININT(Y1,Y2,Z1,Z2,YY,ZZ)  
184      YTEMP(K) = YY  
185      ZTEMP(K) = ZZ  
186 10  CONTINUE  
187  
188      C Renumber the late stations  
189      NSEC = NSEC+1  
190      DO 30 L=NSEC,NPL2+1,-1  
191      X(L) = X(L-1)  
192      DO 20 K=1,KDIM  
193      Y(L,K) = Y(L-1,K)  
194      Z(L,K) = Z(L-1,K)  
195 20  CONTINUE  
196 30  CONTINUE  
197  
198      C Put the temporary array in the grid  
199      DO 50 K=1,KDIM  
200      X(NPL2) = XX  
201      Y(NPL2,K) = YTEMP(K)  
202      Z(NPL2,K) = ZTEMP(K)  
203 50  CONTINUE  
204  
205      RETURN  
206      END
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM  
**samgrid.f**

DATE 7/07/94  
TIME 4:18:56 pm

4

SOURCE TEXT

LINE #

```

207 *****
208 SUBROUTINE CIRCLE(KS,KE,KMAX,Y,Z,RFIL,ARCORR)
209 C DIMENSION X(KMAX),Y(KMAX)
210 C Given a set of pts (Y(i),Z(i)) i=1,...,KMAX, and radius of fillet RFIL,
211 C this subroutine replaces the points (Y(j),Z(j)) j=KS,...,KE by the fillet
212 C points on fillet circle.
213 C Look for the center of the fillet circle (YC,ZC)
214 C YA=Y(KS)
215 C ZA=Z(KS)
216 C YB=Y(KE)
217 C ZB=Z(KE)
218 C
219 C SY=YA-YB
220 C SZ=ZA-ZB
221 C BB=(YA*YA-YB*YB)+(ZA*ZA-ZB*ZB)
222 C R =SY/SZ
223 C
224 C A = 1.*R**2
225 C B = 2.*ZB*R - 2.*YB - BB*R/SZ
226 C C = ZB*ZB+YB*YB + (BB/(2.*SZ))**2 - ZB*BB/SZ - RFIL**2
227 C
228 C DET=B*B-4.*A*C
229 C IF(DET.LE.0.) THEN
230 C   WRITE(15,*)'Determinant is less than 0, ',DET
231 C   GOTO 200
232 C ENDIF
233 C YC1 = ( -B+SQRT(DET) ) / (2.*A)
234 C ZC1 = (BB - 2.*YC1*SY)/(2.*SZ)
235 C YC2 = ( -B-SQRT(DET) ) / (2.*A)
236 C ZC2 = (BB - 2.*YC2*SY)/(2.*SZ)
237 C IF(ZC1.GE.ZC2) THEN
238 C   ZC=ZC1
239 C   YC=YC1
240 C ELSE
241 C   ZC=ZC2
242 C   YC=YC2
243 C ENDIF
244 C
245 C Find the total arc length given
246 C TOTARC=0.
247 DO 50 K=KS,KE-1
248   TOTARC=TOTARC +
249   SQRT( (Y(K+1)-Y(K))**2 + (Z(K+1)-Z(K))**2 )
250 CONTINUE
251 C
252 C Find the arc length b/w two points.
253 C ARC = TOTARC/FLOAT(KE-KS)
254 C ARC = ARC*ARCORR
255 C
256 C
257 C Find the coordinates for each point
258 C DO 100 K=KS,KE-1
259 C   YA=Y(K)
260 C   ZA=Z(K)
261 C   YB=YC
262 C   ZB=ZC
263 C
264 C   SY=YA-YB
265 C   SZ=ZA-ZB
266 C   SR=RFIL**2-ARC**2
267 C   BB=(YA*YA-YB*YB)+(ZA*ZA-ZB*ZB)
268 C   R =SY/SZ
269 C
270 C   A = 1.*R**2
271 C   B = 2.*ZB*R - 2.*YB - BB*R/SZ
272 C   - R*SR/SZ
273 C   C = ZB*ZB+YB*YB + (BB/(2.*SZ))**2 - ZB*BB/SZ - RFIL**2
274 C   + (0.5*SR/SZ)**2 + (BB*SR)/(2.*SZ*SZ) - SR*ZB/SZ
275 C
276 C   DET=B*B-4.*A*C
277 C   IF(DET.LE.0.) THEN
278 C     WRITE(15,*)'Determinant is less than 0, ',DET
279 C     GOTO 200
280 C   ENDIF
281 C   YC1 = ( -B+SQRT(DET) ) / (2.*A)
282 C   ZC1 = (BB - 2.*YC1*SY + (RFIL**2-ARC**2))/(2.*SZ)
283 C   YC2 = ( -B-SQRT(DET) ) / (2.*A)
284 C   ZC2 = (BB - 2.*YC2*SY + (RFIL**2-ARC**2))/(2.*SZ)
285 C   IF(YC1.GE.Y(K)) THEN
286 C     Z(K+1)=ZC1
287 C     Y(K+1)=YC1
288 C   ELSE
289 C     Z(K+1)=ZC2
290 C     Y(K+1)=YC2
291 C   ENDIF
292 C
293 C 100 CONTINUE
294 C 200 RETURN
295 C
296 C

```

UNIX™  
FORTRAN Program

SOURCE PROGRAM  
**samgrid.f**

DATE	7/07/94	PAGE #
TIME	4:18:56 pm	5

LINE #

SOURCE TEXT

```

297 C*****SUBROUTINE CSPLINE(X,Y,N,XNEW,YNEW,NNEW)
298 C
299 C PARAMETER (NMAX=500)
300 C
301 C REAL X(N), Y(N), XNEW(NNEW), YNEW(NNEW)
302 C
303 C ****
304 C
305 C THIS SUBROUTINE PRODUCES A MONOTONE CUBIC SPLINE INTERPOLANT
306 C TO THE DATA (X(I),Y(I)) I=1,...,N AND COMPUTES VALUES AT
307 C THE NEW POINTS XNEW(I), I=1,...,NNEW. THESE ARE RETURNED IN
308 C ARRAY YNEW(I). THE ALGORITHM USED IS THAT OUTLINED BY FRITSCH AND
309 C BUTLAND IN SIAM J. SCI. STAT. COMPUT., VOL. 5, NO. 2, JUNE, 1984.
310 C
311 C
312 C... WRITTEN BY JEFF CORDOVA 10/26/86
313 C
314 C ****
315 C
316 C REAL D(NMAX), DEL(NMAX), H(NMAX)
317 C
318 C ****
319 C SPLINE COEFFICIENT CALCULATIONS
320 C ****
321 C
322 C.... MESH SPACING AND FIRST DIVIDED DIFFERENCE
323 C
324 DO 100 I=1,N-1
325 H(I) = X(I+1) - X(I)
326 100 CONTINUE
327 C
328 DO 200 I=1,N-1
329 DEL(I) = (Y(I+1) - Y(I)) / H(I)
330 200 CONTINUE
331 C
332 C.... SPLINE COEFFICIENTS
333 C
334 C *** LINEAR INTERPOLATION FOR N=2 CASE ***
335 C
336 IF (N .EQ. 2) THEN
337 D(1) = DEL(1)
338 D(N) = DEL(1)
339 GO TO 399
340 ENDIF
341 C
342 C *** MONOTONE SPLINE COEFFICIENTS FOR N >= 3 CASE ***
343 C
344 C.... FIRST BOUNDARY POINT (USE THREE POINT FORMULA ALTERED TO BE
345 C SHAPE PRESERVING)
346 C
347 E$UM = H(1) + H(2)
348 W1 = (H(1) + E$UM) / E$UM
349 W2 = -H(1) / E$UM
350 D(1) = W1*DEL(1) + W2*DEL(2)
351 IF (PCHST(D(1),DEL(1)) .LE. 0.) THEN
352 D(1) = 0.
353 ELSEIF (PCHST(DEL(1),DEL(2)) .LT. 0.) THEN
354 DMAX = 3.*DEL(1)
355 IF (ABS(D(1)) .GT. ABS(DMAX)) D(1) = DMAX
356 ENDIF
357 C
358 C.... INTERIOR POINTS (BRODIE MODIFICATION OF BUTLAND FORMULA)
359 C
360 CONST = 1. / 3.
361 DO 300 I=2,N-1
362 TOP = DEL(I-1) * DEL(I)
363 TOP = TOP + 5. * (1. + SIGN(1.,TOP))
364 ALPHA = CONST * (H(I-1) + 2.*H(I)) / (H(I-1) + H(I))
365 BOT = ALPHA * DEL(I) + (1.-ALPHA) * DEL(I-1) + 1.E-20
366 D(I) = TOP / BOT
367 300 CONTINUE
368 C
369 C.... LAST BOUNDARY POINT (USE THREE POINT FORMULA ADJUSTED TO BE
370 C SHAPE PRESERVING)
371 C
372 E$UM = H(N-2) + H(N-1)
373 W1 = -H(N-1) / E$UM
374 W2 = (H(N-1) + E$UM) / E$UM
375 D(N) = W1*DEL(N-2) + W2*DEL(N-1)
376 IF (PCHST(D(N),DEL(N-1)) .LE. 0.) THEN
377 D(N) = 0.
378 ELSEIF ( PCHST(DEL(N-2),DEL(N-1)) .LT. 0.) THEN
379 DMAX = 3.*DEL(N-1)
380 IF (ABS(D(N)) .GT. ABS(DMAX)) D(N) = DMAX
381 ENDIF
382 C
383 399 CONTINUE
384 C
385 C ****
386 C SPLINE EVALUATION
387 C ****
388 C
389 C.... XNEW(I) .LE. X(N)
390 C
391 IEND = 1
392 DO 400 J=1,N-1
393 CTHREE = (D(J) + D(J+1) - 2.*DEL(J)) / (H(J)*H(J))
394 CTWO = (3.*DEL(J) - 2.*D(J) - D(J+1)) / H(J)
395 IBEG = IEND
396 CRAY : IEND = ISRCMGE(NNEW,XNEW,1,X(J+1)) !OD CRAY
397 IEND = ISRCGE(NNEW,XNEW,1,X(J+1)) !OD WK
398 DO 500 I=IBEG,IEND-1
399 T = XNEW(I) - X(J)
400 YNEW(I) = Y(J) + T*(D(J) + T*(CTWO + T*CTHREE))
401 500 CONTINUE
400 CONTINUE
403 C
404 C.... XNEW(I) .GT. X(N)
405 C
406 DO 600 I=IEND,NNEW
407 T = XNEW(I) - X(N-1)
408 YNEW(I) = Y(N-1) + T*(D(N-1) + T*(CTWO + T*CTHREE))
409 600 CONTINUE
410 C
412 RETURN
413 END

```

## SOURCE TEXT

LINE #

```
414 FUNCTION PCHST(ARG1,ARG2)
415 C ..... ::::::::::::::::::::: ::::::::::::::::::::: :::::::::::::::::::::
416 PCHST = SIGN(1.,ARG1) * SIGN(1.,ARG2)
417 IF ((ARG1.EQ.0.) .OR. (ARG2.EQ.0.)) PCHST = 0.
418 C
419 RETURN
420 END
```

UNIX™  
FORTRAN ProgramSOURCE PROGRAM  
**samgrid.f**DATE 7/07/94 PAGE #  
TIME 4:18:56 pm 7

LINE #

```
422 FUNCTION ISRCHGE(N,X,INCX,FTARGET)
423 DIMENSION X(*)
424 IF(N.LE.0) THEN
425   ISRCHGE = 0
426   RETURN
427 ELSE
428   IT = 1 + (N-1) * INCX
429   ISRCHGE = 1
430   DO 10 I=1,IT,INCX
431     IF(X(I).GE.FTARGET) GOTO 11
432     ISRCHGE = ISRCHGE + 1
433 10  CONTINUE
434 11  CONTINUE
435 ENDIF
436 RETURN
437 END
```

SOURCE TEXT

UNIX™  
FORTRAN ProgramSOURCE PROGRAM  
samgrid.fDATE 7/07/94  
TIME 4:18:56 pm

8

## SOURCE TEXT

LINE #	
438	*****
439	***** SUBROUTINE CUSTER
440	***** include "agrid.com"
441	***** DIMENSION YWK(NPI),ZWK(NPI)
442	***** L2 = L-1
443	***** NFUS = KDIM - NPTS
444	***** NBOT = NEUS/2 + 1
445	***** NTOP = NEUS/2 + 1
446	C DO 900 LL=ML,L2
447	C Note: I am leaving the nose and the wake above
448	C IF(ABS(Y(LL,NBOT)-Y(LL,KDIM-NTOP+1))>E-7) THEN
449	C RETURN
450	C ENDIF
451	C
452	C Do the bottom first:
453	C
454	DO 10 K=1,NBOT
455	YINT(K) = Z(LL,K)
456	ZINT(K) = Y(LL,K)
457	CONTINUE
458	PGSP = SQRT((Z(LL,NBOT)-Z(LL,NBOT+1))**2 +
459	(Y(LL,NBOT)-Y(LL,NBOT+1))**2 )
460	CALL DISTARC(YINT,ZINT,NBOT,YWK,ZWK,NBOT,PGSP,1)
461	DO 80 K=1,NBOT
462	Y(LL,K) = ZWK(K)
463	Z(LL,K) = YWK(K)
464	CONTINUE
465	80
466	C Do the top now:
467	C
468	DO 100 K=1,NTOP
469	YINT(K) = Y(LL,K+(KDIM-NTOP))
470	ZINT(K) = Z(LL,K+(KDIM-NTOP))
471	CONTINUE
472	N1 = (KDIM-NTOP)
473	N2 = (KDIM-NTOP)-1
474	PGSP = SQRT((Z(LL,N1)-Z(LL,N2))**2 +
475	(Y(LL,N1)-Y(LL,N2))**2 )
476	CALL DISTARC(YINT,ZINT,NBOT,YWK,ZWK,NTOP,PGSP,0)
477	DO 180 K=1,NTOP
478	Y(LL,K+(KDIM-NTOP)) = YWK(K)
479	Z(LL,K+(KDIM-NTOP)) = ZWK(K)
480	CONTINUE
481	180
482	C
483	900 CONTINUE
484	RETURN
485	END

LINE #

SOURCE TEXT

```

486 ****
487      SUBROUTINE DISTARC(X,Y,N,XNEW,YNEW,NNEW,FGS,IFLAT)
488
489      DIMENSION X(N),Y(N),XNEW(NNEW),YNEW(NNEW)
490
491      This program redistribute the points (X,Y) by subroutine DISTRI
492      based on the arc length.  FGS is the first grid spacing. Note that
493      the end points of the two sets are the same.
494      IFLAT=0, grid points will cluster near the first point, =1 near the end.
495      Input array is (X(1),Y(1)), i=1,...,N
496      Output array is (XNEW(i),YNEW(i)), i=1,...,NNEW
497
498      C
499      PARAMETER (MAX=400)
500      DIMENSION S(MAX),TOTARC(MAX),XN(MAX),YN(MAX)
501
502      Maximum number of points allowed is MAX
503      IF(MAX.LE.N .OR. MAX.LE.NNEW) THEN
504          WRITE(*,*) 'SUB DISTARC : MAX is less than N or NNEW'
505          STOP
506      ENDIF
507
508      C Look for total arc length
509      TOTARC(1) = 0.
510      DO 10 K=2,N
511          ARC = SQRT( (X(K)-X(K-1))**2 + (Y(K)-Y(K-1))**2 )
512          TOTARC(K) = TOTARC(K-1) + ARC
513 10    CONTINUE
514
515      C Apply subroutine DISTRI to obtain the stretching function S
516      For FGS<0, equal spacing is used
517      IF(FGS.GT.0.) THEN
518          DELT=FGS/TOTARC(N)
519          CALL DISTRI(DELT,NNEW,S,IFLAT)
520      ELSE
521          S(1) = 0.
522          DO 25 K=2,NNEW
523              S(K) = S(K-1) + 1./FLOAT(NNEW-1)
524 25      CONTINUE
525      ENDIF
526
527      C Redistribution, put new array in a temporary arrays XN and YN
528      XN(1)=X(1)
529      YN(1)=Y(1)
530      XN(NNEW)=X(N)
531      YN(NNEW)=Y(N)
532
533      DO 60 J = 2,NNEW
534          ARCNEN = S(J)*TOTARC(N)
535          DO 55 K = 2,N
536              IF(ABS(TOTARC(K)-ARCNEN).LE.1.E-7) THEN
537                  XN(J) = X(K)
538                  YN(J) = Y(K)
539                  GOTO 60
540              ENDIF
541              IF(TOTARC(K).GT.ARCNEN) THEN
542                  X1 = X(K-1)
543                  X2 = X(K)
544                  Y1 = Y(K-1)
545                  Y2 = Y(K)
546                  XX = X1 + (X(K)-X(K-1))*(
547                      (ARCNEN-TOTARC(K-1))/(TOTARC(K)-TOTARC(K-1)))
548                  CALL LININT(X1,X2,Y1,Y2,XX,YY)
549                  XN(J) = XX
550                  IF(ABS(X1-X2).LE.1.E-7) THEN
551                      YN(J) = Y1 + (ARCNEN-TOTARC(K-1))
552                  ELSE
553                      YN(J) = YY
554                  ENDIF
555                  GOTO 60
556              ENDIF
557 55    CONTINUE
558 60    CONTINUE
559
560      C Write the temporary arrays into the output XNEW, YNEW
561      DO 70 J=1,NNEW
562          XNEW(J) = XN(J)
563          YNEW(J) = YN(J)
564 70    CONTINUE
565      RETURN
566  END

```

LINE #

SOURCE TEXT

```
568 ****
569 SUBROUTINE DISTRI(FANG,KFCS,S,IPINE)
570 PARAMETER (MAX=400)
571 DIMENSION S(MAX),DUM(MAX)
572 C
573 C....Calculating the stretching function S when given
574 C....the first spacing, FANG, and the number of points KFCS
575 C....If IPINE=1, distribution is cusltering at outer grid
576 C
577 IF(MAX.LE.KFCS) THEN
578   WRITE(*,*)"SUB-DISTRI : MAX is less than KFCS"
579   STOP
580 ENDIF
581
582 IF(KFCS.EQ.1) THEN
583   S(1) = 0.
584   GOTO 40
585 ENDIF
586 C
587 DZ1 = FANG
588 KFM = KFCS-1
589 DZETA = 1./FLOAT(KFM)
590 RDBETA = 1.5
591 CALL GRBET(DZ1,KFM,0.0001,100,RDBETA)
592 CALL FZ1(KFCS,RDBETA,DZETA,S)
593 C
594 IF (IPINE.EQ.1) THEN
595   DO 37 K=1,KFCS
596     DUM(KFCS-K+1) = S(K)
597   37 CONTINUE
598   DO 38 K=1,KFCS
599     S(K) = 1.-DUM(K)
600   38 CONTINUE
601 ENDIF
602 40 CONTINUE
603 RETURN
604 END
```

LINE #

SOURCE TEXT

```
605 *****
606      SUBROUTINE EDGE(NC,NU,NL,XL,XBK,XBASE,YBASE,ZBASE,NPK,LS)
607      DIMENSION ZBASE(NPK,2,LS),XBASE(NPK,2,LS),YBASE(NPK,2,LS)
608
609      ZLE = ZBASE(1,1,1)
610      DO 200 K =1,NC
611      IF(XBASE(1,1,K).GT.XBK) THEN
612          X1 = XBASE(1,1,K-1)
613          X2 = XBASE(1,1,K )
614          Z1 = ZBASE(1,1,K-1)
615          Z2 = ZBASE(1,1,K )
616          CALL LININT(X1,X2,Z1,Z2,XBK,ZBK)
617          GOTO 210
618      ENDIF
619      200 CONTINUE
620      210 CONTINUE
621
622      C
623      DO 500 K=1,NC
624      IF(ZBASE(1,1,K).GT.ZBK) GOTO 700
625      CALL LININT(ZLE,ZBK,XL,XBK,ZBASE(1,1,K),XLE)
626      XLEOLD = XBASE(1,1,K)
627      XTL = XBASE(NL,1,K)
628      DO 280 I=1,NL
629          F = XBASE(I,1,K)-XLEOLD
630          E = XTL-XBASE(I,1,K)
631          XBASE(I,1,K) = (F*XTL + E*XLE)/(F+E)
632      280 CONTINUE
633      XTL = XBASE(NU,1,K)
634      DO 300 I=1,NU
635          F = XBASE(I,2,K)-XLEOLD
636          E = XTL-XBASE(I,2,K)
637          XBASE(I,2,K) = (F*XTL + E*XLE)/(F+E)
638      300 CONTINUE
639      500 CONTINUE
640      700 CONTINUE
641      RETURN
642      END
```

LINE #

SOURCE TEXT

```
*****  
643 SUBROUTINE EQSPACE  
644 include "agrid.com"  
645 L2 = L-1  
646 DO 130 LL=1,L2  
647 XIN(LL) = X(LL)  
648 DO 120 K=1,KDIM  
649 ZIN(LL,K) = Z(LL,K)  
650 YIN(LL,K) = Y(LL,K)  
651 120 CONTINUE  
652 130 CONTINUE  
653  
654 XTOT = X(L2)-X(1)  
655 DX = XTOT/FLOAT(L2-1)  
656 DO 160 JL=2,L2  
657 X(JL) = X(JL-1)+DX  
658 DO 150 KL=1,L2  
659 IF(ABS(XIN(KL)-X(JL)) .LE. 1.E-7) THEN  
660 DO 140 K=1,KDIM  
661 Z(XL,K) = ZIN(JL,K)  
662 Y(XL,K) = YIN(JL,K)  
663 140 CONTINUE  
664 GOTO 160  
665  
666 ENDIF  
667 IF(XIN(KL).GT.X(JL)) THEN  
668 DO 145 K=1,KDIM  
669 X1 = XIN(KL-1)  
670 X2 = XIN(KL)  
671 Y1 = YIN(KL-1,K)  
672 Y2 = YIN(KL,K)  
673 Z1 = ZIN(KL-1,K)  
674 Z2 = ZIN(KL,K)  
675 XX = X(JL)  
676 CALL LININT(X1,X2,Y1,Y2,XX,YY)  
677 CALL LININT(X1,X2,Z1,Z2,XX,ZZ)  
678 Y(JL,K) = YY  
679 Z(JL,K) = ZZ  
680 145 CONTINUE  
681 GOTO 160  
682  
683 ENDIF  
684 150 CONTINUE  
685 160 CONTINUE  
686 RETURN  
687 END
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM  
**samgrid.f**

DATE 7/07/94  
TIME 4:18:56 pm

PAGE # 13

LINE #

SOURCE TEXT

```

687 ****
688 SUBROUTINE FILET(Y,Z,KDIM,MB1,MB2,MT1,MT2,RFIL)
689 PARAMETER (MAX=400)
690 DIMENSION Z(KDIM),Y(KDIM)
691 DIMENSION D1(MAX),D2(MAX)
692 COMMON /REF/ ZROOT,KTIP,ARCORR
693
C This subroutine takes a wing-fuselage station, (Y(k),Z(k)), k=1,KDIM,
694 and find the two (top and bottom) intersections of the wing and the
695 fuselage.
696 And then, for example, at the bottom intersection (Y(k),Z(k)), it
697 extends to a segment of points, (Y(k),Z(k)), k=KF1 to KF2, where
698 KF1=K-MB1, KF2=K+MB2.
699 Similar procedure for the top part.
700 And then, call subroutine CIRCLE to replace the segment by a segment
701 of a circle with radius RFIL.
702
703
704 IF(RFIL.EQ.0.) GOTO 735
705
706 IF(MAX.LE.KDIM) THEN
707   WRITE(*,*)'SUB FILET : MAX is less than KDIM'
708   STOP
709 ENDIF
710
711
712 Bottom part of the aircraft
713 IF(MB1.EQ.0 .AND. MB2.EQ.0) GOTO 135
714 DO 130 K=1,KDIM
715   IF (K.LE.KTIP .AND. Z(K).GT.ZROOT) THEN
716     KF1=K-MB1
717     KF2=K+MB2
718     CALL CIRCLE(KF1,KF2,KDIM,Y,Z,RFIL,ARCORR)
719
720   We have N=KF2-KF1+1 pts in fillet area, employ two more points
721   from the original grid and redistribute them, the grid spacing looks
722   smoother.
723   KF1=KF1+1
724   KF2=KF2+1
725   DO 220 KD=KF1,KF2
726     D1(KD-KF1+1) = Y(KD)
727     D2(KD-KF1+1) = Z(KD)
728   CONTINUE
729   N=KF2-KF1+1
730   CALL DISTARC(D2,D1,N,D2,D1,N,-10.,0)
731   DO 330 KD=KF1,KF2
732     Y(KD)=D1(KD-KF1+1)
733     Z(KD)=D2(KD-KF1+1)
734   CONTINUE
735   GOTO 135
736
737 ENDIF
738 130 CONTINUE
739 135 CONTINUE
740
741 Top part of the aircraft
742 IF(MT1.EQ.0 .AND. MT2.EQ.0) GOTO 735
743 DO 700 K=KTIP,KDIM
744   IF (K.GT.KTIP .AND. Z(K).LE.ZROOT) THEN
745     KF1=K-MT2
746     KF2=K+MT1
747     CALL CIRCLE(KF1,KF2,KDIM,Y,Z,RFIL,ARCORR)
748
749   We have N=KF2-KF1+1 pts in fillet area, employ two more points
750   from the original grid and redistribute them, the grid spacing looks
751   smoother.
752   KF1=KF1+1
753   KF2=KF2+1
754   DO 420 KD=KF1,KF2
755     D1(KD-KF1+1) = Y(KD)
756     D2(KD-KF1+1) = Z(KD)
757   CONTINUE
758   N=KF2-KF1+1
759   CALL DISTARC(D1,D2,N,D1,D2,N,-10.,0)
760   DO 530 KD=KF1,KF2
761     Y(KD)=D1(KD-KF1+1)
762     Z(KD)=D2(KD-KF1+1)
763   CONTINUE
764   GOTO 735
765
766 ENDIF
767 700 CONTINUE
768 735 CONTINUE
769
770 RETURN
771 END

```

## LINE #

## SOURCE TEXT

```
770 *****
771 SUBROUTINE FZ1(L1,TBETA,DET,Z)
772 C
773 C COMPUTES NORMALIZED NORMAL DISTANCE, Z(L)
774 C
775 C PARAMETER (MAX=400)
776 C DIMENSION Z(MAX)
777
778 IF(MAX.LE.L1) THEN
779   WRITE(*,*) 'SUB FZ1 : MAX is less than L1'
780   STOP
781 ENDIF
782
783 IF(TBETA.EQ.1.) THEN
784   DO 10 L=1,L1
785   Z(L)=0.
786 10 CONTINUE
787 ELSE
788   DO 20 L=1,L1
789   ETA=(L-1)*DET
790   RR=(TBETA+1.)/(TBETA-1.)
791   EEE=1.-ETA
792   RBETA=RR**EEE
793   Z(L)=(TBETA-1.)*(RR-RBETA)/(RBETA+1.)
794 20 CONTINUE
795 END IF
796 RETURN
797 END
```

LINE #

SOURCE TEXT

```
798 ****  
799      SUBROUTINE GRBET(DFM,NPT,FPCC,ICC,BETA)  
800      C  
801      BISECTION METHOD USED TO DETERMINE STRETCHING PARAMETER, BETA,  
802      WHICH GIVES DESIRED GY AT THE WALL  
803      C  
804      PARAMETER (MAX=400)  
805      DIMENSION Z(MAX)  
806      IF(MAX.LE.NPT) THEN  
807          WRITE(*,*)'SUB GRBET : MAX is less than NPT'  
808          STOP  
809      ENDIF  
810      C  
811      ICCL=ICC  
812      FPCCCL=FPCC*DFM  
813      BETAI=BETA  
814      Z1=DFM  
815      DET=1./NPT  
816      BR=1.  
817      FR=-Z1  
818      IICC=ICC/10  
819      DO 10 I=1,IICC  
820      BF=BETAI  
821      BETA=0.5*(BETAI+1.)  
822      CALL F21(2,BF,DET,Z)  
823      FF=Z(2)-Z1  
824      IF(FF.GT.0.) GO TO 15  
825      BETAI=2.*BETAI-1.  
826      10 CONTINUE  
827      15 CONTINUE  
828      DO 5 NIT=1,ICCL  
829      CALL F21(2,BETA,DET,Z)  
830      F=Z(2)-Z1  
831      IF(F.GT.0.) THEN  
832          FF=F  
833          BF=BETA  
834          ELSE  
835              FR=F  
836              BR=BETA  
837          END IF  
838          BETA=0.5*(BF+BR)  
839          IF(ABS(F).LT.FPCCCL) GO TO 4  
840      5 CONTINUE  
841      WRITE(6,100) BETA,F  
842      100 FORMAT(1B0,36H EXCEEDED MAX. NO. OF ITS...BETA,F ,3G13.6)  
843      4 CONTINUE  
844      C      CALL F21(2,BETA,DET,Z)  
845      C      F=Z(2)-Z1  
846      BM1=BETA-1.  
847      BFM1=BF-1.  
848      BRM1=BR-1.  
849      RETURN  
850      END
```

## LINE #

## SOURCE TEXT

```
851 *****
852      SUBROUTINE LININT(X1,X2,Y1,Y2,XLOCAL,YLOCAL)
853      C This subroutine linearly interpolate YLOCAL when given (X1,Y1) & (X2,Y2)
854      IF(ABS(X1-X2).LE.1.E-7) THEN
855          YLOCAL=(Y2+Y1)/2.
856          GOTO 100
857      ENDIF
858      SLOPE = (Y2-Y1)/(X2-X1)
859      YLOCAL = SLOPE*(XLOCAL-X2) + Y2
860      CONTINUE
861      RETURN
862      END
```

LINE # SOURCE TEXT

```

863 *****SUBROUTINE MOUNT(JN,LNUM,IPRNT)
864      SUBROUTINE MOUNT(JN,LNUM,IPRNT)
865      include "sgrid.com"
866      DIMENSION NPAIR(2,NPI),LNUM(4)
867      DIMENSION YWK(NPI),ZWK(NPI),YNAC(NPI),ZNAC(NPI)
868      DIMENSION YNG(2*NPI),ZNG(2*NPI)
869      COMMON /ENG/ XENG(2,NPI),YENG(2,NPI,NPI),ZENG(2,NPI,NPI)
870           ,MNAC(2),MNACP(2)
871
872      C   (XENG,YENG,ZENG)    Coordinate of engine
873      C   MNAC(*)             Number of stations in nacelle
874      C   MNACP(*)            Number of points in each station
875      C   (1,*,* ) inner nacelle, (2,*,* ) outer nacelle
876      C
877      C   MC      *   # of pts added in the grid (*=# pts at nacelle)
878      C   IPRNT   =   0 no writing out
879      C
880      MC = 40
881      NPWN = NPTS+MC
882      IF(LNUM(1).EQ.LNUM(3) .AND. LNUM(2).EQ.LNUM(4)) THEN
883          NPTS = NPTS + MC
884          NPWN = NPTS + MC
885      ENDIF
886      NPB = (NPTS+1)/2
887      NPWNH = (NPWN+1)/2
888      NNAC = MNAC(JN)
889      MNACP = MNACP(JN)
890
891      IF(IPRNT.NE.0) WRITE(IPRNT)NPWN,1,LNUM(2)-LNUM(1)+1
892
893      Now the big job!
894
895      DO 800 L=LNUM(1),LNUM(2)
896
897      C   Interpolate the points of the nacelle at x=xout
898      DO 100 LN=1,NNAC
899          IF(ABS(XENG(JN,LN)-XOUT(L,1)).LE.1.E-7) THEN
900              DO K=1,NNACP
901                  YNAC(K) = YENG(JN,LN,K)
902                  ZNAC(K) = ZENG(JN,LN,K)
903              ENDDO
904              GOTO 105
905          ELSEIF(XENG(JN,LN).GT.XOUT(L,1)) THEN
906              DO K=1,NNACP
907                  X1 = XENG(JN,LN-1)
908                  Y1 = YENG(JN,LN-1,K)
909                  Z1 = ZENG(JN,LN-1,K)
910                  X2 = XENG(JN,LN )
911                  Y2 = YENG(JN,LN ,K)
912                  Z2 = ZENG(JN,LN ,K)
913                  XX = XOUT(L,1)
914                  CALL LININT(X1,X2,Y1,Y2,XX,YY)
915                  CALL LININT(X1,X2,Z1,Z2,XX,ZZ)
916                  YNAC(K) = YY
917                  ZNAC(K) = ZZ
918              ENDDO
919              GOTO 105
920          ENDIF
921      CONTINUE
922      CONTINUE
923      RADNAC = SQRT( (ZNAC(1)-ZNAC>NNAC/2))**2 +
924          (YNAC(1)-YNAC>NNAC/2))**2 )
925
926      C   Count the number of points in the wake if we are in the wake
927      KOUNT = 0
928      DO 120 K=1,NPB-1
929          K1=NPH-K
930          K2=NPH+K
931          IF(ABS(YOUT(L,K1)-YOUT(L,K2)).LE.1.E-7 .AND.
932              ABS(ZOUT(L,K1)-ZOUT(L,K2)).LE.1.E-7 ) THEN
933              KOUNT = KOUNT + 1
934              NPAIR(1,KOUNT) = K1
935              NPAIR(2,KOUNT) = K2
936          ENDIF
937      CONTINUE
938
939      C   Nacelle totally under the wing, INTRAIL = 0
940      C   Part under the wing, part in the wake, INTRAIL = 1
941      C   Nacelle totally in the wake, INTRAIL = 2
942      INTRAIL = 0
943      IF(KOUNT.EQ.0) GOTO 415
944      IF(ZOUT(L,NPAIR(1,1)).GT.ZNAC(1 )) INTRAIL=1
945      IF(ZOUT(L,NPAIR(1,1)).GT.ZNAC>NNAC)) INTRAIL=2
946      write(*,*)'INTRAIL = ',INTRAIL
947      IF(INTRAIL.NE.2) THEN
948          We are not in the wake region or the trailing edge
949          GOTO 415
950      ELSE
951          We are in the wake region or the trailing edge
952          WRITE(*,*)'We are in the wake region'
953          NTEMP = 1
954      CONTINUE
955      Obtain all points under the wake line
956      DO 180 K=1,KOUNT
957          IF(ZOUT(L,NPAIR(1,K)).LT.ZNAC(NTEMP)) THEN
958              KS = K
959              r1 = abs(zout(L,NPAIR(1,K))-zout(L,NPAIR(1,K+1)))
960              r2 = abs(zout(L,NPAIR(1,K))-ZNAC(NTEMP))
961              if(r2.le.r1/8.) KS=K1
962              IF(YOUT(L,NPAIR(1,KS)).LT.YNAC(NTEMP)) THEN
963                  NTEMP = NTEMP + 1
964                  GOTO 130
965              ENDIF
966              GOTO 185
967          ENDIF
968      CONTINUE
969      CONTINUE
970      NS = NTEMP
971      IF(NS.NE.1) THEN
972          CALL LININT(YNAC(NS-1),YNAC(NS),ZNAC(NS-1),ZNAC(NS),
973                      YOUT(L,NPAIR(1,KS)),ZZS)
974          YY = YOUT(L,NPAIR(1,KS))
975      ELSE
976          CALL LININT(ZOUT(L,NPAIR(1,KS)),ZOUT(L,NPAIR(1,KS)-1),
977                      YOUT(L,NPAIR(1,KS)),YOUT(L,NPAIR(1,KS)-1),
978                      ZNAC(NS),YY)
979          ZZS = ZNAC(NS)
980      ENDIF
981      NTEMP = NNACP

```

LINE #	SOURCE TEXT
983	190 CONTINUE
984	IF(INTRAIL.EQ.2) THEN
985	DO 200 K=1,KOUNT
986	IF(ZOUT(L,NPAIR(1,K)).LT.ZNAC(NTEMP)) THEN
987	KE = K-1
988	IF(YOUT(L,NPAIR(1,KE)).LT.ZNAC(NTEMP)) THEN
989	NTEMP = NTEMP - 1
990	GOTO 190
991	ENDIF
992	GOTO 215
993	ENDIF
994	200 CONTINUE
995	CONTINUE
996	NE = NTEMP
997	IF(NTEMP.NE.NNACP) THEN
998	CALL LININT(YNAC(NE-1),ZNAC(NE),ZNAC(NE-1),ZNAC(NE),
999	YOUT(L,NPAIR(1,KE)),ZZE)
000	ELSE
001	CALL LININT(ZOUT(L,NPAIR(1,KE)),ZOUT(L,NPAIR(1,KE)+1),
002	YOUT(L,NPAIR(1,KE)),YOUT(L,NPAIR(1,KE)+1),
003	ZNAC(NE),YYE)
004	ZZE = ZNAC(NE)
005	ENDIF
006	ELSE
007	DO 230 K=1,NPH
008	IF(ZOUT(L,K).GT.ZNAC(NTEMP)) THEN
009	KE = K
010	IF(YOUT(L,KE).LT.ZNAC(NTEMP)) THEN
011	NTEMP = NTEMP - 1
012	GOTO 190
013	ENDIF
014	GOTO 235
015	ENDIF
016	CONTINUE
017	CONTINUE
018	NE = NTEMP
019	CALL LININT(ZOUT(L,KE),ZOUT(L,KE-1),YOUT(L,KE),YOUT(L,KE-1),
020	ZNAC(NE),YYE)
021	ZZE = ZNAC(NE)
022	ENDIF
023	C Form the lower surface
024	C Store wingroot grid
025	DO K=1,NPAIR(1,KS)
026	YNNG(K) = YOUT(L,K)
027	ZNNG(K) = ZOUT(L,K)
028	ENDDO
029	C Store wingtip grid to a temp array
030	IF(INTRAIL.EQ.2) THEN
031	DO K=NPAIR(1,KE),NPH
032	YINT(K-NPAIR(1,KE)+1) = YOUT(L,K)
033	ZINT(K-NPAIR(1,KE)+1) = ZOUT(L,K)
034	ENDDO
035	ELSE
036	DO K=KE,NPH
037	YINT(K-KE+1) = YOUT(L,K)
038	ZINT(K-KE+1) = ZOUT(L,K)
039	ENDDO
040	ENDIF
041	C Get the nacelle grid under the wake line ready
042	DO K=NS,NE
043	YNK(K-NS+2) = YNAC(K)
044	ZWK(K-NS+2) = ZNAC(K)
045	ENDDO
046	IF(INTRAIL.EQ.2) THEN
047	NNACC = NPNHH-NPAIR(1,KS)-(NPH-NPAIR(1,KE)+1)
048	ELSE
049	NNACC = NPNHH-NPAIR(1,KS)-(NPH-KE+1)
050	ENDIF
051	CALL DISTARC(YNK,ZWK,NADD,YNK,ZWK,NNACC,-10,0)
052	C Stick the nacelle grid under the wing
053	DO K=1,NNACC
054	ZNNG(NPAIR(1,KS)+K) = ZWK(K)
055	YNNG(NPAIR(1,KS)+K) = YNK(K)
056	ENDDO
057	C Put back the wingtip grid into the wing
058	IF(INTRAIL.EQ.2) THEN
059	KREP = NPH-NPAIR(1,KE)+1
060	DO K=1,KREP
061	ZNNG(NPAIR(1,KS)+NNACC+K) = ZINT(K)
062	YNNG(NPAIR(1,KS)+NNACC+K) = YINT(K)
063	ENDDO
064	ELSE
065	KREP = NPH-KE+1
066	DO K=1,KREP
067	ZNNG(NPAIR(1,KS)+NNACC+K) = ZINT(K)
068	YNNG(NPAIR(1,KS)+NNACC+K) = YINT(K)
069	ENDDO
070	ENDIF
071	NNNACH = NPNHH
072	C Form the upper surface
073	C Store the wingtip grid
074	IF(INTRAIL.EQ.2) THEN
075	DO K=NPH,NPAIR(2,KE)
076	YNNG(NNNACH+K-NPH) = YOUT(L,K)
077	ZNNG(NNNACH+K-NPH) = ZOUT(L,K)
078	ENDDO
079	ELSE
080	DO K=NPH,NPTS
081	IF(ZOUT(L,K).GT.ZZE) THEN
082	KN1 = K
083	GOTO 260
084	ENDIF
085	ENDDO
086	CONTINUE
087	DO K=NPH,KN1
088	YNNG(NNNACH+K-NPH) = YOUT(L,K)
089	ZNNG(NNNACH+K-NPH) = ZOUT(L,K)
090	ENDDO
091	NI = NNNACH+NPAIR(2,KE)-NPH
092	ENDIF
093	260
094	CONTINUE
095	DO K=NPH,KN1
096	YNNG(NNNACH+K-NPH) = YOUT(L,K)
097	ZNNG(NNNACH+K-NPH) = ZOUT(L,K)
098	ENDDO
099	NI = NNNACH+KN1-NPH
100	
101	
102	

LINE #	SOURCE TEXT
1103	ENDIF
1104	C Store the wingroot grid to a temp array
1105	DO K=NPAIR(2,KS),NPTS
1106	YINT(K-NPAIR(2,KS)+1) = YOUT(L,K)
1107	ZINT(K-NPAIR(2,KS)+1) = ZOUT(L,K)
1108	ENDDO
1109	C Get the nacelle grid above the wake line ready
1110	NADD = NNACP-NE + NS-1 + 2
1111	YWK(1) = YYE
1112	ZWK(1) = ZZE
1113	DO K=NE+1,NNACP
1114	YWK(K-NE+1) = YNAC(K)
1115	ZWK(K-NE+1) =ZNAC(K)
1116	ENDDO
1117	DO K=1,NS-1
1118	YWK(NNACP-NE+1+K) = YNAC(K)
1119	ZWK(NNACP-NE+1+K) =ZNAC(K)
1120	ENDDO
1121	YWK(NADD) = YYS
1122	ZWK(NADD) = ZZE
1123	CALL DISTARC(YWK,ZWK,NADD,YWK,ZWK,NNACC,-10,0)
1124	C Stick the nacelle grid above the wing
1125	DO K=1,NNACC
1126	ZNNG(N1+K) = ZWK(K)
1127	YWNG(N1+K) = YWK(K)
1128	ENDDO
1129	C Put back the wingroot grid into the wing
1130	KREP = NPTS-NPAIR(2,KS)+1
1131	DO K=1,KREP
1132	ZNNG(N1+NNACC+K) = ZINT(K)
1133	YWNG(N1+NNACC+K) = YINT(K)
1134	ENDDO
1135	GOTO 700
1136	ENDIF
1137	C We are before trailing edge, normal redistribution
415	CONTINUE
416	C Bottom part
417	C Find out the points that being replaced by the nacelle
418	NTEMP = 1
419	CONTINUE
420	DO 440 K=1,NPH
421	IF(ZOUT(L,K).GT.ZNAC(NTEMP)) THEN
422	KS = K-1
423	r1 = abs(zout(L,KS)-zout(L,KS-1))
424	r2 = abs(zout(L,KS)-ZNAC(NTEMP))
425	if(r2.le.r1/8.) then
426	KS=KS-1
427	endif
428	IF(ABS(YOUT(L,KS)-YNAC(NTEMP)).LE.RADNAC/800.) THEN
429	NTEMP = NTEMP + 1
430	GOTO 430
431	ENDIF
432	CALL LININT(ZOUT(L,KS),ZOUT(L,KS+1),YOUT(L,KS),
433	YOUT(L,KS+1),ZNAC(NTEMP),YYS)
434	ZZE = ZNAC(NTEMP)
435	GOTO 445
436	ENDIF
437	440 CONTINUE
438	445 CONTINUE
439	NFRIS = NTEMP
440	C
441	NTEMP = NNACP
442	CONTINUE
443	DO 460 K=KS+1,NPH
444	IF(ZOUT(L,K).GT.ZNAC(NTEMP)) THEN
445	KE = K
446	IF(ABS(YOUT(L,KE)-YNAC(NTEMP)).LE.RADNAC/800.) THEN
447	NTEMP = NTEMP - 1
448	GOTO 450
449	ENDIF
450	CALL LININT(ZOUT(L,KE-1),ZOUT(L,KE),YOUT(L,KE-1),
451	YOUT(L,KE),ZNAC(NTEMP),YYE)
452	ZZE = ZNAC(NTEMP)
453	GOTO 465
454	ENDIF
455	460 CONTINUE
456	465 CONTINUE
457	NFRIE = NTEMP
458	C
459	Store wingroot grid
460	DO 500 K=1,KS
461	YWNG(K) = YOUT(L,K)
462	ZNNG(K) = ZOUT(L,K)
463	CONTINUE
464	C Get the nacelle grid under the wake line ready
465	N2 = NFRIE-NFRIS+1 +2
466	YINT(1) = YYS
467	ZINT(1) = ZZE
468	YINT(N2) = YYE
469	ZINT(N2) = ZZE
470	DO 520 K=NFRIS,NFRIE
471	YINT(K-NFRIS+2) = YNAC(K)
472	ZINT(K-NFRIS+2) =ZNAC(K)
473	CONTINUE
474	NNACC = NPNNH-KS-(NPH-KE+1)
475	CALL DISTARC(YINT,ZINT,N2,YWK,ZWK,NNACC,-10,0)
476	C Stick the nacelle grid under the wing
477	DO K=1,NNACC
478	ZNNG(KS+K) = ZWK(K)
479	YWNG(KS+K) = YWK(K)
480	ENDDO
481	C Put the wingtip grid
482	DO 540 K=KE,NPH
483	YWNG(KS+NNACC+K-KE+1) = YOUT(L,K)
484	ZNNG(KS+NNACC+K-KE+1) = ZOUT(L,K)
485	CONTINUE
486	C
487	Upper part
488	DO 590 K=NPH,NPTS
489	IF(ZOUT(L,K).LE.ZNAC(NFRIE)) THEN
490	NN1 = K-1
491	CALL LININT(ZOUT(L,NN1+1),ZOUT(L,NN1),YOUT(L,NN1+1),
492	YOUT(L,NN1),ZNAC(NFRIE),YY1)
493	ZZ1 = ZNAC(NFRIE)
494	GOTO 595
495	ENDIF
496	590 CONTINUE
497	595 CONTINUE

LINE #

SOURCE TEXT

```
1223 DO 600 K=NPE,NPTS
1224   IF(ZOUT(L,K).LE.ZNAC(NFRIS)) THEN
1225     NN2 = K
1226     CALL LININT(ZOUT(L,NN2),ZOUT(L,NN2-1),YOUT(L,NN2),
1227           YOUT(L,NN2-1),ZNAC(NFRIS),YY2)
1228     Z22 = ZNAC(NFRIS)
1229     GOTO 605
1230   ENDIF
1231 600  CONTINUE
1232 605  CONTINUE
1233 C..... Store the wingtip grid
1234 DO 620 I=NPE,NN1
1235   YWNG(NPWNH+K-NPE) = YOUT(L,K)
1236   ZWNG(NPWNH+K-NPE) = ZOUT(L,K)
1237 620  CONTINUE
1238 C..... Redistribute the points above the macelle
1239 N2 = NN2-NN1+1
1240   YWK(1) = YY1
1241   ZWK(1) = ZZ1
1242   YWK(N2) = YY2
1243   ZWK(N2) = ZZ2
1244 DO K=NN1+1,NN2-1
1245   YWK(K-NN1+1) = YOUT(L,K)
1246   ZWK(K-NN1+1) = ZOUT(L,K)
1247 ENDDO
1248 NNA = NPWNH-(NN1-NPE+1)-(NPTS-NN2+1)
1249 CALL DISTARC(YWK,ZWK,N2,YWK,ZWK,NNA,-10,0)
1250 NNT = NPWNH+(NN1-NPE)
1251 DO K=1,NNA
1252   YWNG(NNT+K) = YWK(K)
1253   ZWNG(NNT+K) = ZWK(K)
1254 ENDDO
1255 C..... Store the wingroot grid
1256 DO K=NN2,NPTS
1257   YWNG(NNT+NNA+K-NN2+1) = YOUT(L,K)
1258   ZWNG(NNT+NNA+K-NN2+1) = ZOUT(L,K)
1259 ENDDO
1260 700  CONTINUE
1261 IF(IPRNT.NE.0)THEN
1262   WRITE(IPRNT)(XOUT(L,1),K=1,NPWN),
1263             (YWNG(K),K=1,NPWN),
1264             (ZWNG(K),K=1,NPWN)
1265   CALL FLUSH (IPRNT)
1266 ENDIF
1267
1268 DO 750 K=1,NPWN
1269   XOUT(L,K) = XOUT(L,1)
1270   YOUT(L,K) = YWNG(K)
1271   ZOUT(L,K) = ZWNG(K)
1272 750  CONTINUE
1273 C..... 800  CONTINUE
1274 IF(LNUM(1).EQ.LNUM(3) .AND. LNUM(2).EQ.LNUM(4)) THEN
1275   NPTS = NPTS - NC
1276 ENDIF
1277 RETURN
1278 END
```

LINE #

SOURCE TEXT

```

1280 *****
1281      SUBROUTINE NACGRID
1282      include "agrid.com"
1283      C This subroutine read the nacelle grid and combine it with the
1284      C wing grid.
1285      C
1286      COMMON /ENG/ XENG(2,NPI),YENG(2,NPI,NPI),ZENG(2,NPI,NPI)
1287      ' ,MNAC(2),MNACP(2)
1288      COMMON /NACG/ XNAC(NPI),YNAC(NPI,NPI),ZNAC(NPI,NPI)
1289      COMMON /NDIM/ NNAC,NNACP,LNUM(4)
1290
1291      C (XENG,YENG,ZENG)   Coordinate of engine
1292      C MNAC(*)           Number of stations in nacelle
1293      C MNACP(*)          Number of points in each station
1294      C (1,*,") inner nacelle; (2,*,") outer nacelle
1295      C
1296      C
1297      C Read the nacelles geometry:
1298      C
1299      C Inner nacelle geometry
1300      CALL NACIN
1301      XIN1 = INAC(1)
1302      XIN2 = INAC(NNAC)
1303      MNAC(1) = NNAC
1304      MNACP(1) = NNACP
1305      DO 100 L=1,NNAC
1306      XENG(1,L) = XNAC(L)
1307      DO 80 K=1,NNACP
1308      YENG(1,L,K) = YNAC(L,K)
1309      ZENG(1,L,K) = ZNAC(L,K)
1310      80 CONTINUE
1311      100 CONTINUE
1312      C
1313      C Outer nacelle geometry
1314      CALL NACIN
1315      XOUT1 = XNAC(1)
1316      XOUT2 = XNAC(NNAC)
1317      MNAC(2) = NNAC
1318      MNACP(2) = NNACP
1319      DO 200 L=1,NNAC
1320      XENG(2,L) = XNAC(L)
1321      DO 180 K=1,NNACP
1322      YENG(2,L,K) = YNAC(L,K)
1323      ZENG(2,L,K) = ZNAC(L,K)
1324      180 CONTINUE
1325      200 CONTINUE
1326      C
1327      C
1328      C In a case of 2 nacelles, three zone will be made.
1329      C NEWING : Station(s) will be added to the wing at inlet and outlet
1330      C of the nacelle
1331      C
1332      C MOUNT : Mount the nacelle under the wing and/or wake line
1333      C
1334      C JN = 1    Inner nacelle
1335      C         2    Outer nacelle
1336      C
1337      C
1338      C The first zone, only one nacelle appears
1339      WRITE(*,*)'zone 1'
1340      JN = 1
1341      CALL NEWING(LNUM,XIN1,XOUT1)
1342      CALL MOUNT(JN,LNUM,21)
1343      LNUM(3) = LNUM(1)
1344      LNUM(4) = LNUM(2)
1345
1346      C The second zone consists two nacelles appear
1347      WRITE(*,*)'zone 2'
1348      JN = 1
1349      CALL NEWING(LNUM,XOUT1,XIN2)
1350      LNUM(1) = LNUM(1) + 1
1351      CALL MOUNT(JN,LNUM,0)
1352      LNUM(3) = LNUM(1)
1353      LNUM(4) = LNUM(2)
1354
1355      C
1356      WRITE(*,*)'zone 2'
1357      CALL NEWING(LNUM,XOUT1,XIN2)
1358      LNUM(1) = LNUM(1) + 1
1359      JN = 2
1360      CALL MOUNT(JN,LNUM,22)
1361      LNUM(3) = LNUM(1)
1362      LNUM(4) = LNUM(2)
1363
1364      C The third zone, only one nacelle appears
1365      WRITE(*,*)'zone 3'
1366      JN = 2
1367      CALL NEWING(LNUM,XIN2,XOUT2)
1368      LNUM(1) = LNUM(1) + 1
1369      CALL MOUNT(JN,LNUM,23)
1370
1371      RETURN
END

```

LINE #

SOURCE TEXT

```
*****  
1372 SUBROUTINE NEWING(LNUM,XX1,XX2)  
1373 include "agrid.com"  
1375 DIMENSION XWNG(NPI,NPI),YWNG(PPI,NPI),ZWNG(NPI,NPI)  
1376 DIMENSION LNUM(4)  
1377  
1378 C Rewrite the coordinate of the wing  
1379 DO 10 L=1,NSEC  
1380 DO 10 K=1,NPTS  
1381 XWNG(L,K) = XOUT(L,K)  
1382 YWNG(L,K) = YOUT(L,K)  
1383 ZWNG(L,K) = ZOUT(L,K)  
1384 10 CONTINUE  
1385  
1386 C Find out where the x-location of start and end of nacelle  
1387 XNSTRT = XX1  
1388 XNEND = XX2  
1389  
1390  
1391 C Add two stations in the wing, these two stations lie exactly on  
1392 C XNSTRT and XNEND  
1393 DO 55 NN=1,2  
1394 IF(NN.EQ.1) XX=XNSTRT  
1395 IF(NN.EQ.2) XX=XNEND  
1396 DO 50 LW=1,NSEC  
1397 C The wing section is very close to nacelle's station (XNSTRT or XNEND)  
1398 IF(ABS(XWNG(LW,1)-XX).LE.1.E-7) THEN  
1399 DO K=1,NPTS  
1400 XWNG(LW,K) = XX  
1401 ENDDO  
1402 LNUM(NN) = LW  
1403 GOTO 55  
1404 ENDIF  
1405 C Create an extra station in the wing  
1406 IF(XWNG(LW,1).GT.XX) THEN  
1407 X(LW) = XX  
1408 LNUM(NN) = LW  
1409 DO K=1,NPTS  
1410 X1 = XWNG(LW-1,K)  
1411 Y1 = YWNG(LW-1,K)  
1412 Z1 = ZWNG(LW-1,K)  
1413 X2 = XWNG(LW ,K)  
1414 Y2 = YWNG(LW ,K)  
1415 Z2 = ZWNG(LW ,K)  
1416 CALL LININT(X1,X2,Y1,Y2,XX,YY)  
1417 CALL LININT(X1,X2,Z1,Z2,XX,ZZ)  
1418 Y(LW,K) = YY  
1419 Z(LW,K) = ZZ  
1420 ENDDO  
1421 C If we are in wake, make sure top pts intersect the bottom pts  
1422 DO 24 K1=2,(NPTS+1)/2  
1423 DO 20 K2=(NPTS+1)/2+1,NPTS  
1424 IF(ABS(Y(LW,K1)-Y(LW,K2)).LT.1.E-6 .AND.  
1425 ABS(Z(LW,K1)-Z(LW,K2)).LT.1.E-6 ) THEN  
1426 Y(LW,K1) = Y(LW,K2)  
1427 Z(LW,K1) = Z(LW,K2)  
1428 GOTO 24  
1429 ENDIF  
1430 20 CONTINUE  
1431 24 CONTINUE  
1432 C Put the rest of the station into (X,Y,Z)  
1433 DO 35 L=LW,NSEC  
1434 DO 30 K=1,NPTS  
1435 X(L+1) = XWNG(L,K)  
1436 Y(L+1,K) = YWNG(L,K)  
1437 Z(L+1,K) = ZWNG(L,K)  
1438 30 CONTINUE  
1439 35 CONTINUE  
1440 NSEC = NSEC + 1  
1441 DO 45 L=LW,NSEC  
1442 DO 40 K=1,NPTS  
1443 XWNG(L,K) = X(L)  
1444 YWNG(L,K) = Y(L,K)  
1445 ZWNG(L,K) = Z(L,K)  
1446 40 CONTINUE  
1447 45 CONTINUE  
1448 GOTO 55  
1449 ENDIF  
1450 50 CONTINUE  
1451 55 CONTINUE  
1452  
1453 DO 910 L=1,NSEC  
1454 DO 910 K=1,NPTS  
1455 XOUT(L,K) = XWNG(L,K)  
1456 YOUT(L,K) = YWNG(L,K)  
1457 ZOUT(L,K) = ZWNG(L,K)  
1458 910 CONTINUE  
1459 RETURN  
1460 END
```

LINE #

SOURCE TEXT

```
1461 ****
1462 SUBROUTINE NOSE(FNBOT,FNTOP)
1463 include "sgrid.com"
1464 DIMENSION D1(NPI),D2(NPI),S(NPI)
1465
1466 KTIP = (KDIM+1)/2
1467
1468 C From the nose to the leading edge
1469 C ie., from station 1 to M1
1470
1471 C First point of the nose
1472 DO 12 K=1,KDIM
1473   X(1) = XIN(1)
1474   Y(1,K) = YIN(1,1)
1475   Z(1,K) = ZIN(1,1)
1476 12 CONTINUE
1477
1478 C Loop for all stations, from station 2 to station M1
1479 DO 500 M=2,M1
1480   I = M
1481   XLOCAL = XIN(M)
1482   Store the input to dummy arry
1483   DO 33 K=1,NPI
1484     YINT(K) = YIN(M,K)
1485     ZINT(K) = ZIN(M,K)
1486 33 CONTINUE
1487
1488 C YREF is value of the first point of the wing.
1489 C KREF is the corresponding index of each station.
1490 C YREF=YOUT(1,NPTS)
1491
1492 C OPTIONS :-
1493
1494 C Boeing Baseline Configuration
1495 DO 43 K=1,NPI
1496   IF(YINT(K).GE.YREF) THEN
1497     KREF=K
1498     GOTO 44
1499   ENDIF
1500 43 CONTINUE
1501 44 CONTINUE
1502   KREF=NPI/2 + KREFADD
1503
1504 C Langley's Low Boom Configuration
1505 C KREF = 31
1506
1507
1508 C Lower part of the nose (-Y to Y=YREF)
1509 KS=1
1510 KE=KREF
1511 KN=KE-KS+1
1512 DO 74 K= KS,KE
1513   KK = K-KS+1
1514   D1(KK) = YINT(K)
1515   D2(KK) = ZINT(K)
1516 74 CONTINUE
1517 CALL DISTARC(D1,D2,KN,YNEW,ZDIST,KTIP,FNBOT,1)
1518 DO 85 K=1,KTIP
1519   Y(L,K) = YNEW(K)
1520   Z(L,K) = ZDIST(K)
1521   X(L) = XLOCAL
1522 85 CONTINUE
1523
1524 C From Y=YREF to pos Y
1525 KS=KREF
1526 KE=NPI
1527 KN=KE-KS+1
1528 DO 100 K= KS,KE
1529   KK = K-KS+1
1530   D1(KK) = YINT(K)
1531   D2(KK) = ZINT(K)
1532 100 CONTINUE
1533 CALL DISTARC(D1,D2,KN,YNEW,ZDIST,KTIP,FNTOP,0)
1534 DO 400 K=KTIP,KDIM
1535   Y(L,K) = YNEW(K-KTIP+1)
1536   Z(L,K) = ZDIST(K-KTIP+1)
1537   X(L) = XLOCAL
1538 400 CONTINUE
1539 500 CONTINUE
1540 RETURN
1541
1542 END
```

## SOURCE TEXT

```

LINE #
1543 *****
1544      SUBROUTINE REDIST(XLOCAL,KTIP,FAC,IFLAT)
1545      include "sgrid.com"
1546
1547      C This is the main subroutine to redistribute the points from
1548      C spanwise cut to streamwise cut.
1549      C When there is "cheung-SIMP", it means the output is for SIMP code.
1550      C
1551      PARAMETER(INT=400)
1552      DIMENSION S(INT)
1553
1554      C NU = number of point in the upper surface
1555      C NL = number of point in the lower surface
1556      C NC = number of spanwise sections
1557      C XLE = 1 leading edge
1558      C (ZINT,YINT) = point of streamwise cut for X=XLOCAL at each surface
1559      C KT = # of pts in circum. direction extracted from the old grid.
1560      C KTIP = # of pts in the circum. direction in one surface.
1561
1562      C IFLAT = 2 : upper surface
1563      C IFLAT = 1 : lower surface
1564
1565      IF(INT.LE.KTIP) THEN
1566          WRITE(*,*)'SUB REDIST : INT is less than KTIP'
1567          STOP
1568      ENDIF
1569
1570      IF(IFLAT.EQ.2) THEN
1571          NUL = NU
1572      ELSE
1573          NUL = NL
1574      ENDIF
1575
1576      C The streamwise distance passes the leading edge
1577      IF(XLOCAL.GT.XLE(NC))THEN
1578
1579      C X-station is at the wing tip
1580      IF(XLOCAL.LE.(XLE(NC)+CHORD(NC))) THEN
1581
1582      C Should call WINGNNAKE if wake is contained in this station
1583      IF(XLOCAL.GT.XLE(1)+CHORD(1)) THEN
1584          CALL WINGNNAKE(XLOCAL,KT,NUL,IFLAT)
1585          GOTO 200
1586
1587      ENDIF
1588
1589      C Streamwise distance is between the leading edge and the trailing edge
1590      DO 60 M=1,NC
1591      DO 50 I=3,NUL
1592
1593      C     IF(XBASE(I,IFLAT,NC-M+1).GE.XLOCAL .OR.
1594          ABS(XBASE(I,IFLAT,NC-M+1)-XLOCAL).LE.1.E-7) THEN
1595          X1 = XBASE(I-1,IFLAT,NC-M+1)
1596          X2 = XBASE(I,IFLAT,NC-M+1)
1597          Y1 = YBASE(I-1,IFLAT,NC-M+1)
1598          Y2 = YBASE(I,IFLAT,NC-M+1)
1599          Z1 = ZBASE(I-1,IFLAT,NC-M+1)
1600          Z2 = ZBASE(I,IFLAT,NC-M+1)
1601          CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
1602          CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
1603          YINT(M) = YY
1604          ZINT(M) = ZZ
1605          GOTO 60
1606
1607      ENDIF
1608      50 CONTINUE
1609      60 CONTINUE
1610      KT=NC
1611      GOTO200
1612
1613      C ELSE
1614
1615      C     The X-station passes the wing tip, should have wake
1616      CALL WINGNNAKE(XLOCAL,KT,NUL,IFLAT)
1617      GOTO 200
1618
1619      C Streamwise distance is in the leading edge
1620      IF(XLOCAL.LE.XLE(NC)) THEN
1621
1622      C     Should call WINGNNAKE if wake is contained in this station
1623      IF(XLOCAL.GT.XLE(1)+CHORD(1)) THEN
1624          CALL WINGNNAKE(XLOCAL,KT,NUL,IFLAT)
1625          GOTO 200
1626
1627      ENDIF
1628
1629      KT = 0
1630      DO 160 I = 1,NUL
1631          KT = KT + 1
1632
1633      C Create a point at the root
1634      IF(XBASE(I,IFLAT,1).GT.XLOCAL) THEN
1635          X1 = XBASE(I-1,IFLAT,1)
1636          X2 = XBASE(I,IFLAT,1)
1637          Y1 = YBASE(I-1,IFLAT,1)
1638          Y2 = YBASE(I,IFLAT,1)
1639          Z1 = ZBASE(I-1,IFLAT,1)
1640          Z2 = ZBASE(I,IFLAT,1)
1641          CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
1642          CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
1643          YINT(KT) = YY
1644          ZINT(KT) = ZZ
1645          GOTO 200
1646
1647      ENDIF
1648      DO 150 M = 2,NC
1649          XL = XLOCAL-XBASE(I,IFLAT,M-1)
1650          XR = XLOCAL-XBASE(I,IFLAT,M)
1651          IF(XL*XR.LT.0.) THEN
1652              X1 = XBASE(I,IFLAT,M-1)
1653              X2 = XBASE(I,IFLAT,M)
1654              Z1 = ZBASE(I,IFLAT,M-1)
1655              Z2 = ZBASE(I,IFLAT,M)
1656              Y1 = YBASE(I,IFLAT,M-1)
1657              Y2 = YBASE(I,IFLAT,M)
1658              CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
1659              CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
1660              YINT(KT) = YY
1661              ZINT(KT) = ZZ
1662              GOTO 160
1663
1664      ELSEIF (ABS(XL*XR) .LE. 1.E-7) THEN
1665          IF(ABS(XL).LE.1.E-7) THEN

```

LINE #	SOURCE TEXT
1663	YINT(KT) = YBASE(I,IFLAT,M-1)
1664	ZINT(KT) = ZBASE(I,IFLAT,M-1)
1665	ELSE
1666	YINT(KT) = YBASE(I,IFLAT,M)
1667	ZINT(KT) = ZBASE(I,IFLAT,M)
1668	ENDIF
1669	GOTO 160
1670	ENDIF
1671	CONTINUE
1672	CONTINUE
1673	ENDIF
1674	c
1675	200 CONTINUE
1676	c
1677	Right now YINT and ZINT is from wing tip to the root, in order to
1678	the cubic spline program, need from root to tip.
1679	c
1680	DO 220 KK=1,KT
1681	S(KK) = YINT(KK)
1682	CONTINUE
1683	DO 230 KK=1,KT
1684	YINT(KK) = S(KT-KK+1)
1685	CONTINUE
1686	DO 240 KK=1,KT
1687	S(KK) = ZINT(KK)
1688	CONTINUE
1689	DO 250 KK=1,KT
1690	ZINT(KK) = S(KT-KK+1)
1691	250 CONTINUE
1692	c
1693	c OPTIONS :-
1694	c Distribute z coordinates (rspe around direction) from root to
1695	c leading edge and back (xdist)
1696	c IT=1, grid points will cluster near the wing tip, ~0 near the root.
1697	c For HESS, IT=0 , for Boeing, IT=1
1698	IT=1
1699	CALL DISTARC(ZINT,YINT,KT,ZDIST,YNEW,KTIP,PAC,IT)
1700	IF(I.NE.NSEC)CALL CSPLINE(ZINT,YINT,KT,ZDIST,YNEW,KTIP)
1701	c
1702	RETURN
1703	END

## SOURCE TEXT

```
LINE # *****  
1704      SUBROUTINE SUBTRACGRID(NPLI,X,Y,Z,NPI,NSEC,KDIM)  
1705  
1706 C This subroutine allows us to subtract a grid line NPLI in  
1707 C streamwise section, and the new dimension is NSEC again.  
1708 C  
1709 C  
1710      DIMENSION X(NPI),Y(NPI,NPI),Z(NPI,NPI)  
1711 C  
1712 C     Renumber the late stations  
1713      NSEC = NSEC-1  
1714      DO 30 L=NPLI,NSEC  
1715      X(L) = X(L+1)  
1716      DO 20 K=1,KDIM  
1717      Y(L,K) = Y(L+1,K)  
1718      Z(L,K) = Z(L+1,K)  
1719      20 CONTINUE  
1720      30 CONTINUE  
1721  
1722      RETURN  
1723      END
```

LINE #

SOURCE TEXT

```

1724 ****
1725 SUBROUTINE TAIL(FAC1,FAC2)
1726 include "agrid.com"
1727 DIMENSION S(NPI)
1728 COMMON /REF/ ZROOT,KTIP,ARCORR
1729
1730 KTIP = (KDIM+1)/2
1731 LE = L-1
1732 C The tail of the configuration
1733 C WAKEPT is the Y value that the wake is (ie Y value of ZROOT)
1734 C
1735 PT1= (YOUT(NSEC,1)+YOUT(NSEC,NPTS))/2.
1736 PT2= (YIN(NF,NFP)+YIN(NF,1))/2.
1737 X1 = X(LE)
1738 X2 = XIN(M)
1739
1740 L1 = LE+1
1741 L2 = L1+(NF-M2)-1
1742 DO 300 L=L1,L2
1743   M=M2+L-L1+1
1744   X(L) = XIN(M)
1745   XLOCAL = XIN(M)
1746 C OPTIONS :-
1747 C
1748 C   WAKEPT = (YIN(NF,NFP)+YIN(NF,1))/2.
1749 C
1750 CALL LININT(X1,X2,PT1,PT2,XLOCAL,WAKEPT)
1751 C
1752 C Design the 1st point (KF) will be the off fuselage side
1753 C It is also the number of pts in the upper or lower fuselage, therefore
1754 C it depends on the previous station.
1755 C (no. of pt in this station)/(no. of pts in previous) =
1756 C (radius of this station) // radius of previous
1757 C RATIO=ABS(YIN(M,1)-YIN(M,NFP))/ABS(YIN(M-1,1)-YIN(M-1,NFP))
1758 KF=(KDIM-NPTS)/2+1
1759 KF=(KDIM-NPTS)/2
1760 IF(RATIO.LE.0.9 .OR. RATIO.GE.1.1) THEN
1761 KF=IFIX(RATIO*FLOAT(KF))
1762 ENDIF
1763 C
1764 C Find KT1 the point at old grid where YIN(M,KT1)=WAKEPT
1765 C and calculate the points from neg Y to Y at KT1
1766 DO 222 K=1,NFP
1767   IF(ABS(YIN(M,K)-WAKEPT).LE.1.E-7) THEN
1768     KT1=K
1769     GOTO 223
1770   ENDIF
1771 IF(YIN(M,K).GT.WAKEPT) THEN
1772   KT1=K
1773   Y1=YIN(M,K-1)
1774   Y2=YIN(M,K)
1775   Z1=ZIN(M,K-1)
1776   Z2=ZIN(M,K)
1777   YY=WAKEPT
1778   CALL LININT(Y1,Y2,Z1,Z2,YY)
1779   YIN(M,KT1)=YY
1780   ZIN(M,KT1)=ZZ
1781   GOTO 223
1782 ENDIF
1783
1784 CONTINUE
1785 222 CONTINUE
1786 223 CONTINUE
1787
1788 DO 230 K=1,KT1
1789   YINT(K)=YIN(M,K)
1790   ZINT(K)=ZIN(M,K)
1791 230 CONTINUE
1792 CALL DISTARC(YINT,ZINT,KT1,YNEW,ZDIST,KF,FAC1,1)
1793 DO 240 K=1,KF
1794   Y(L,K)=YNEW(K)
1795   Z(L,K)=ZDIST(K)
1796 240 CONTINUE
1797 C
1798 C Find KT2 the point at old grid where YIN(M,KT2)=WAKEPT
1799 C note : KT2=NFP-KT1+1
1800 C and calculate the points from neg Y to Y at KT1
1801 C KT2=NFP-KT1+1
1802 DO 250 K=1,KT2
1803   YINT(K)=YIN(M,KT1+K-1)
1804   ZINT(K)=ZIN(M,KT1+K-1)
1805 250 CONTINUE
1806 CALL DISTARC(YINT,ZINT,KT2,YNEW,ZDIST,KF,FAC1,0)
1807 DO 260 K=1,KF
1808   Y(L,KDIM-KF+K)=YNEW(K)
1809   Z(L,KDIM-KF+K)=ZDIST(K)
1810 260 CONTINUE
1811 C
1812 C These are points in off fuselage side
1813 C KN=No. of pts in the off-fuselage side
1814 C KN=KDIM-2-KF
1815 C KN=(KN+1)/2
1816 C RSPAN= ABS( Z(LE,KTIP)-Z(L,KF) )
1817 C IF(FAC2.LT.0.)FAC2=1.E+15
1818 C DELT=FAC2*(RSPAN)/FLOAT(KNB)
1819 C CALL DISTRI(DELT,KNB+1,S,0)
1820 C
1821 CALL LININT(X1,X2,YOUT(NSEC,NPTS/2),PT2,XLOCAL,YOB)
1822 C
1823 DO 270 K=1,KNH
1824   Z(L,KF+K)=Z(L,KF)+S(K+1)*RSPAN
1825   CALL LININT(ZROOT,RSPAN,WAKEPT,YOB,Z(L,KF+K),YY)
1826   Y(L,KF+K)=YY
1827 270 CONTINUE
1828 DO 280 K=1,KNH
1829   Z(L,KDIM-KF-K+1)=Z(L,KF+K)
1830   Y(L,KDIM-KF-K+1)=Y(L,KF+K)
1831 280 CONTINUE
1832 C Smooth the fuselage-wake part
1833 DO 285 KK=1,KDIM
1834   YINT(KK)=Y(L,KK)
1835   ZINT(KK)=Z(L,KK)
1836 285 CONTINUE
1837 ZROOT = ZIN(M,KT2)
1838 RFILT = RFIL
1839 IF(X(L).GE.XSTART .AND. X(L).LE.XOFF)
1840   CALL FILET(YINT,ZINT,KDIM,MB1,MB2,MT1,MT2,RFILT)
1841
1842
1843

```

LINE #

SOURCE TEXT

```
1844 DO 290 K=1,EDIM
1845 Y(L,K)=YINT(K)
1846 Z(L,K)=ZINT(K)
1847 290 CONTINUE
1848 cheung-SIMP
1849 c      NPH=(NPP+1)/2
1850 c      DO 293 K=1,NPH
1851 c      YINT(K)=YIN(M,K)
1852 c      ZINT(K)=ZIN(M,K)
1853 293 CONTINUE
1854 CALL DISTARC(YINT,ZINT,NPH,YNEW,ZDIST,KTIP,0.05,1)
1855 DO 295 K=1,KTIP
1856 c      Z(L,K)=ZDIST(K)
1857 c      Y(L,K)=YNEW(K)
1858 295 CONTINUE
1859 c      DO 296 K=1,NPH
1860 c      YINT(K)=YIN(M,NPH+K-1)
1861 c      ZINT(K)=ZIN(M,NPH+K-1)
1862 296 CONTINUE
1863 CALL DISTARC(YINT,ZINT,NPH,YNEW,ZDIST,KTIP,0.05,0)
1864 DO 297 K=1,KTIP
1865 c      Z(L,KTIP+K-1)=ZDIST(K)
1866 c      Y(L,KTIP+K-1)=YNEW(K)
1867 297 CONTINUE
1868 cheung-SIMP
1869 300 CONTINUE
1870 c      this allows the tail has equal thickness
1871 c      DO 313 L=L1+1,L2
1872 c      DO 312 K=1,EDIM
1873 c      Z(L,K)=Z(L-1,K)
1874 c      Y(L,K)=Y(L-1,K)
1875 c 313 CONTINUE
1876 c 313 CONTINUE
1877 c
1878 RETURN
1879 END
```





## LINE # SOURCE TEXT

```
2068 ****
2069 SUBROUTINE WFMATCH
2070 include "sgrid.com"
2071 DIMENSION D1(NPI),D2(NPI)
2072 COMMON /REF/ ZROOT,KTIP,ARCORE
2073 C
2074 C This subroutine moves the whole inward or outward such that the
2075 C fuselage and the wing-root match.
2076 C
2077 C
2078 C The wing-body section, itself has NSEC sections
2079 C LW is the section index for YIN,ZIN
2080 DZMAX = 0.
2081 DZMIN = 1.E+20
2082 LB = M1+1
2083 LE = M1+(NSEC-1)
2084 DO 200 L=LB,LE
2085 LW=L-M1+1
2086 XLOCAL = XOUT(LW,1)
2087 C Calculate the points (YINT,ZINT) on the fuselage at XLOCAL
2088 C Note: assumed that in this motion, each station has same number
2089 C of points in rope-around direction.
2090 C
2091 DO 10 M=M1,NF
2092 IF(XIN(M).GE.XLOCAL .OR.
2093 & ABS(XIN(M)-XLOCAL).LE.1.E-7) THEN
2094 MW = M
2095 GOTO 15
2096 ENDIF
2097 10 CONTINUE
2098 15 CONTINUE
2099 DO 30 K=1,NFP
2100 X1=XIN(MW-1)
2101 X2=XIN(MW)
2102 Y1=YIN(MW-1,K)
2103 Y2=YIN(MW,K)
2104 Z1=ZIN(MW-1,K)
2105 Z2=ZIN(MW,K)
2106 CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
2107 CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
2108 ZINT(K) = ZZ
2109 YINT(K) = YY
2110 30 CONTINUE
2111 35 CONTINUE
2112 C Move the wing in the z-direction (spanwise) to make sure
2113 C the wing is not inside or outside the fuselage
2114 ZROOT = ZINT((NFP+1)/2)
2115 DO KZ1 = 1,NFP
2116 IF(ZINT(KZ1).GT.ZROOT) ZROOT=ZINT(KZ1)
2117 ENDDO
2118 DZ = ZOUT(LW,1)-ZROOT
2119 IF(ABS(DZ).GT.ABS(DZMAX)) DZMAX=DZ
2120 IF(ABS(DZ).LT.ABS(DZMIN)) DZMIN=DZ
2121 200 CONTINUE
2122 IF(DZMAX.GT.0.) DZ=DZMIN
2123 IF(DZMIN.LT.0.) DZ=DZMAX
2124 DO 400 K=1,NPTS
2125 DO 400 L=1,NSEC
2126 ZOUT(L,K) = ZOUT(L,K) - DZ
2127 400 CONTINUE
2128 write(*,*)'DZ =', DZ
2129 RETURN
2130 END
```

LINE #

SOURCE TEXT

```

2132 ****SUBROUTINE WING_BODY
2133 include "sgrid.com"
2134 DIMENSION D1(NPI),D2(NPI)
2135 COMMON /REF/ ZROOT,RTIP,ARCORR
2136
2137
2138 C The wing-body section, itself has NSEC sections
2139 C LW is the section index for YIN,ZIN
2140 C
2141 C LB = M1+1
2142 C LE = M1 + NSEC -1
2143 DO 200 L=LB,LE
2144 C LW=L-M1+1 !start with second wing station
2145 XLOCAL = XOUT(LW,1)
2146
2147 C Calculate the points (YINT,ZINT) on the fuselage at XLOCAL
2148 C Note: assumed that in this section, each station has same number
2149 C of points in rope-around direction.
2150 C
2151 DO 10 M=M1,NF
2152 IF(XIN(M).GE.XLOCAL .OR.
2153 ABS(XIN(M)-XLOCAL).LE.1.E-7) THEN
2154 MW = M
2155 GOTO 15
2156 ENDIF
2157 10 CONTINUE
2158 15 CONTINUE
2159 DO 30 K=1,NFP
2160 X1=XIN(MW-1)
2161 X2=XIN(MW)
2162 Y1=YIN(MW-1,K)
2163 Y2=YIN(MW,K)
2164 Z1=ZIN(MW-1,K)
2165 Z2=ZIN(MW,K)
2166 CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
2167 CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
2168 ZINT(K) = ZZ
2169 YINT(K) = YY
2170 30 CONTINUE
2171 35 CONTINUE
2172
2173 C There are NPTS points in the wing area, need to check out
2174 C how many point is in the fuselage section.
2175 C K1 is the inters pt ./ the fuselage & wing at bottom in old grid
2176 C K2 is the inters pt ./ the fuselage & wing at top in old grid
2177 C MID is the inter point ./ the fuselage & wing at bottom in new grid
2178 C There are NT points from 1 to K1
2179 C
2180 KT=(NFP+1)/2
2181 MID=(KDIM-NPTS+2)/2
2182 MID=(KDIM-NPTS)/2
2183
2184 C Find K1
2185 DO 50 K=KT,1,-1
2186 IF(YINT(K).LT.YOUT(LW,1) .AND. YINT(K).NE.YINT(K-1)) THEN
2187 K1 = K
2188 GOTO 52
2189 ENDIF
2190 50 CONTINUE
2191 52 CONTINUE
2192
2193 C Find K2
2194 DO 70 K=1,NFP
2195 IF(YINT(K).GT.YOUT(LW,NPTS) .AND. YINT(K).NE.YINT(K+1)) THEN
2196 K2 = K
2197 GOTO 72
2198 ENDIF
2199 70 CONTINUE
2200 72 CONTINUE
2201 C For arrow-wing type, K2 needed to be relocated
2202 IF(ABS(YOUT(LW,1)-YOUT(LW,NPTS)).LE.1.E-7) THEN
2203 K1 = K1 + KIADD
2204 K2 = K1
2205 YINT(K2)=YINT(K1)
2206 ZINT(K2)=ZINT(K1)
2207 ENDIF
2208
2209 C Calculate the points at the bottom (from neg. Y to Y at MID)
2210 KS=1
2211 KE= K1
2212 KN=KE-KS+1
2213 DO 82 K= KS,KE
2214 KK = K-KS+1
2215 D1(KK) = YINT(K)
2216 D2(KK) = ZINT(K)
2217 82 CONTINUE
2218 CALL DISTARC(D1,D2,KN,YNEW,ZDIST,MID,-10.,1)
2219 DO 100 K=1,MID
2220 Y(L,K) = YNEW(K)
2221 Z(L,K) = ZDIST(K)
2222 X(L) = XLOCAL
2223 100 CONTINUE
2224 C The points of the wing section
2225 DO 110 K=1,NPTS
2226 Y(L,MID+K) = YOUT(LW,K)
2227 Z(L,MID+K) = ZOUT(LW,K)
2228 X(L) = XOUT(LW,K)
2229 110 CONTINUE
2230
2231 C Calculate the points at the top
2232 KS=K2
2233 KE= NFP
2234 KN=KE-KS+1
2235 DO 115 K= KS,KE
2236 KK = K-KS+1
2237 D1(KK) = YINT(K)
2238 D2(KK) = ZINT(K)
2239 115 CONTINUE
2240 CALL DISTARC(D1,D2,KN,YNEW,ZDIST,MID,-10.,0)
2241 DO 120 K=1,MID
2242 Y(L,K+MID+NPTS) = YNEW(K)
2243 Z(L,K+MID+NPTS) = ZDIST(K)
2244 X(L) = XLOCAL
2245 120 CONTINUE
2246 C For arrow-wing type, make sure wake points ok
2247 IF(ABS(YOUT(LW,1)-YOUT(LW,NPTS)).LE.1.E-7) THEN
2248 Y(L,MID+NPTS+1) = YOUT(LW,1)
2249 Y(L,MID) = Y(L,MID+NPTS+1)
2250 2251

```

LINE #

SOURCE TEXT

```
2252      Z(L,MID) = Z(L,MID+NPTS+1)
2253      ENDIF
2254
2255      C
2256      C      Fill the unsmooth part by FILET
2257      C      First of all, find the set of points needed to be rearrange
2258      DO 125 KK=1,KDIM
2259          YINT(KK)=Y(L,KK)
2260          ZINT(KK)=Z(L,KK)
2261 125      CONTINUE
2262      RFILT = RFIL
2263      IF(X(L).GE.XSTART .AND. X(L).LE.XOFF)
2264      CALL FILET(YINT,ZINT,KDIM,MB1,MB2,MT1,MT2,RFILT)
2265      DO 140 K=1,KDIM
2266          Y(L,K)=YINT(K)
2267          Z(L,K)=ZINT(K)
2268 140      CONTINUE
2269 200      CONTINUE
2270      RETURN
2271      END
```

LINE #	SOURCE TEXT
2272	*****
2273	SUBROUTINE WINGWNWAKE(XLOCAL,KT,NUL,IFLAT)
2274	include "sgrid.com"
2275	C This subroutine creates the data when the station is in the place
2276	where some points are on the wing, some are the wake.
2277	At that x-station, we put 10 points in the wake part.
2278	C
2279	KT = 0.
2280	IF(ARRWING.GT.0.) THEN
2281	The wing is sweeped backwards
2282	DO 75 J=2,NC
2283	J1=J-1
2284	J2=J
2285	IF(XLOCAL.GT.XLE(J)+CHORD(J)) GOTO 75
2286	X11=XLE(J1)+CHORD(J1)
2287	Z1=ZBASE(NUL,IFLAT,J1)
2288	X12=XLE(J2)+CHORD(J2)
2289	Z2=ZBASE(NUL,IFLAT,J2)
2290	ZTIP=Z1+((XLOCAL-X11)/(X12-X11))*(Z2-Z1)
2291	ZFLAT=ZTIP
2292	C
2293	NLEDG = NC
2294	DO MRUN=2,NC
2295	IF(XLOCAL.LT.XBASE(1,IFLAT,MRUN)) THEN
2296	If XLOCAL < leading edge, we should do the following
2297	NLEDG = MRUN-1
2298	X1 = XBASE(1,IFLAT,MRUN-1)
2299	X2 = XBASE(1,IFLAT,MRUN )
2300	Y1 = YBASE(1,IFLAT,MRUN-1)
2301	Y2 = YBASE(1,IFLAT,MRUN )
2302	Z1 = ZBASE(1,IFLAT,MRUN-1)
2303	Z2 = ZBASE(1,IFLAT,MRUN )
2304	CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
2305	CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
2306	YINT(1) = YY
2307	ZINT(1) = ZZ
2308	GOTO 40
2309	ENDIF
2310	ENDDO
2311	CONTINUE
2312	40
2313	DO 65 M=NLEDG,J2,-1
2314	DO 64 I=2,NUL
2315	IF(XLOCAL.LE.XBASE(I,IFLAT,M)) THEN
2316	X1 = XBASE(I-1,IFLAT,M)
2317	X2 = XBASE(I ,IFLAT,M)
2318	Y1 = YBASE(I-1,IFLAT,M)
2319	Y2 = YBASE(I ,IFLAT,M)
2320	Z1 = ZBASE(I-1,IFLAT,M)
2321	Z2 = ZBASE(I ,IFLAT,M)
2322	CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
2323	CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
2324	IF(NLEDG.EQ.NC) THEN
2325	MM=NLEDG-M+1
2326	ELSE
2327	MM=NLEDG-M+2
2328	ENDIF
2329	YINT(MM) = YY
2330	ZINT(MM) = ZZ
2331	GOTO 65
2332	ENDIF
2333	CONTINUE
2334	64
2335	CONTINUE
2336	C Add 10 points for the wake.
2337	ZROOT = ZBASE(NUL,IFLAT,1)
2338	Y1=YBASE(NUL,IFLAT,J1)
2339	Y2=YBASE(NUL,IFLAT,J2)
2340	Z1=ZBASE(NUL,IFLAT,J1)
2341	Z2=ZBASE(NUL,IFLAT,J2)
2342	CALL LININT(X1,X12,Y1,Y2,XLOCAL,YY)
2343	CALL LININT(X1,X12,Z1,Z2,XLOCAL,ZZ)
2344	DZ = (ZFLAT-ZROOT)/9.
2345	DO 70 M=1,10
2346	IF(NLEDG.EQ.NC) THEN
2347	MM = NLEDG-J2+1+M
2348	ELSE
2349	MM = NLEDG-J2+1+M +1
2350	ENDIF
2351	ZINT(MM) = ZROOT + FLOAT(10-M)*DZ
2352	CALL LININT(ZROOT,Z2,YOUT(L-1,1),YY,ZINT(MM),YYY)
2353	YINT(MM) = YYY
2354	70
2355	CONTINUE
2356	IF(NLEDG.EQ.NC) THEN
2357	KT = (NLEDG-J2+1)+10
2358	ELSE
2359	KT = (NLEDG-J2+1 +1)+10
2360	ENDIF
2361	GOTO 200
2362	75
2363	CONTINUE
2364	C
2365	The wing is sweeped forwards
2366	DO 105 J=2,NC
2367	J1=J-1
2368	J2=J
2369	IF(XLOCAL.LT.XLE(J)+CHORD(J)) GOTO 105
2370	X11=XLE(J1)+CHORD(J1)
2371	Z1=ZBASE(NUL,IFLAT,J1)
2372	X12=XLE(J2)+CHORD(J2)
2373	Z2=ZBASE(NUL,IFLAT,J2)
2374	ZTIP=Z1+((XLOCAL-X11)/(X12-X11))*(Z2-Z1)
2375	ZFLAT=ZTIP
2376	cheung-SIMP c     ZTIP=ZBASE(NUL,IFLAT,NC)
2377	ZTIP=ZBASE(NUL,IFLAT,NC)
2378	cheung
2379	C
2380	Add 10 points for the wake.
2381	Y1=YBASE(NUL,IFLAT,J1)
2382	Y2=YBASE(NUL,IFLAT,J2)
2383	Z1=ZBASE(NUL,IFLAT,J1)
2384	Z2=ZBASE(NUL,IFLAT,J2)
2385	CALL LININT(X1,X12,Y1,Y2,XLOCAL,YY)
2386	CALL LININT(X1,X12,Z1,Z2,XLOCAL,ZZ)
2387	DZ = (ZTIP-ZFLAT)/9.
2388	DO 80 M=1,10
2389	ZINT(M) = ZTIP - FLOAT(M-1)*DZ
2390	CALL LININT(ZTIP,Z2,YOUT(L-1,NPTS/2+1),YY,ZINT(M),YYY)
2391	

LINE #

SOURCE TEXT

```
2392      YINT(M) = YYY
2393 80    CONTINUE
2394  DO 100 M=J1,1,-1
2395  DO 90 I=2,NUL
2396  IF(XLOCAL.LE.XBASE(I,IFLAT,M)) THEN
2397    X1 = XBASE(I-1,IFLAT,M)
2398    X2 = XBASE(I ,IFLAT,M)
2399    Y1 = YBASE(I-1,IFLAT,M)
2400    Y2 = YBASE(I ,IFLAT,M)
2401    Z1 = ZBASE(I-1,IFLAT,M)
2402    Z2 = ZBASE(I ,IFLAT,M)
2403    CALL LININT(X1,X2,Y1,Y2,XLOCAL,YY)
2404    CALL LININT(X1,X2,Z1,Z2,XLOCAL,ZZ)
2405    YINT(J1-M+1 +10) = YY
2406    ZINT(J1-M+1 +10) = ZZ
2407    GOTO 100
2408  ENDIF
2409 90    CONTINUE
2410 100   CONTINUE
2411   IT = 10 + J1
2412   GOTO200
2413 105   CONTINUE
2414  ENDIF
2415 C
2416 200   CONTINUE
2417   RETURN
2418  END
```

LINE #

SOURCE TEXT

```
2419 C*****  
2420 SUBROUTINE WINGIN  
2421 include "sgrid.com"  
2422 C  
2423 C NC = # of sections in the spanwise direction  
2424 C ZBASE = Z value of Kth section  
2425 C XLE(K) = leading edge X value of Kth section  
2426 C YLE(K) = leading edge Y value of Kth section  
2427 C CHORD(K) = Chord length of the Kth section  
2428 C NU = # of points in the upper & lower (Kth) section  
2429 C Note : the end points of upper and lower sections are same physical pts  
2430 C  
2431 C MACH2 configuration:  
2432 READ(10,900)  
2433 READ(10,910)  
2434 READ(10,920)NC,NU  
2435 NL = NU  
2436  
2437 K=1  
2438 111 CONTINUE  
2439 READ(10,930)  
2440 READ(10,950)  
2441 DO 12 I=1,NU  
2442 12 READ(10,*)XBASE(I,2,K),YBASE(I,2,K),ZBASE(I,2,K)  
2443 READ(10,940)  
2444 READ(10,950)  
2445 DO 15 I=1,NU  
2446 15 READ(10,*)XBASE(I,1,K),YBASE(I,1,K),ZBASE(I,1,K)  
2447 XLE(K) = XBASE(I,1,K)  
2448 YLE(K) = YBASE(I,1,K)  
2449 CHORD(K) = ABS(XBASE(I,1,K)-XBASE(NU,1,K))  
2450 K=K+1  
2451 IF(K .LT. NC)GOTO111  
C  
2452 900 FORMAT(1X)  
2453 910 FORMAT(1X)  
2454 920 FORMAT(25X,I5,9X,I5/)  
2455 930 FORMAT(1X)  
2456 940 FORMAT(1X)  
2457 950 FORMAT(1X)  
2458 960 FORMAT(3P16.7)  
2459 RETURN  
2460 END
```

UNIX™  
FORTRAN ProgramSOURCE PROGRAM  
**samgrid.f**DATE 7/07/94  
TIME 4:18:56 pm PAGE # 37

LINE #	SOURCE TEXT
2463	SUBROUTINE FUSEIN
2464	include "sgrid.com"
2465	DIMENSION YT(NPI),ZT(NPI)
2466	C
2467	C
2468	NP = # of section in the fuselage
2469	NPP = # of points in mth section
2470	C
2471	C
2472	Read the fuselage geometry
2473	C
2474	MACH2 configuration
2475	READ(10,810)
2476	READ(10,820)
2477	READ(10,830)NF,NPP
2478	C
2479	M=0
2480	40 CONTINUE
2481	C
2482	READ(10,840)
2483	M = M + 1
2484	C
2485	DO 100 K=1,NPP
2486	READ(10,*) XIN(M),YIN(M,K),ZIN(M,K)
2487	YT(K)=YIN(M,K)
2488	ZT(K)=ZIN(M,K)
2489	100 CONTINUE
2490	C CALL DISTARC(YT,ET,NPP,YT,ZT,NPP,-10.,0)
2491	DO 110 K=1,NPP
2492	YT(M,K)=YT(K)
2493	ZIN(M,K)=ZT(K)
2494	110 CONTINUE
2495	IF(M.LT.NF) GOTO 40
2496	C Write the fuselage geometry into PLOT3D Planar format
2497	C
2498	KW=1
2499	KK=NPP
2500	WRITE(51)KK,KW,M
2501	DO 800 L=1,M
2502	WRITE(51)(XIN(L),K=1,KK),
2503	(YIN(L,K),K=1,KK),
2504	(ZIN(L,K),K=1,KK)
2505	800 CONTINUE
2506	C
2507	810 FORMAT(1X)
2508	820 FORMAT(1X)
2509	830 FORMAT(3IX,15.8I,15/)
2510	840 FORMAT(1X)
2511	850 FORMAT(2X,3F16.7)
2512	RETURN
2513	END

LINE #

SOURCE TEXT

```
2514 SUBROUTINE NACIN
2515 include "sgrid.com"
2516 DIMENSION YT(NPI), ZT(NPI)
2517 COMMON /NACC/ XNAC(NPI), YNAC(NPI,NPI),ZNAC(NPI,NPI)
2518 COMMON /NDIM/ NNAC,NNACP,INUM(4)
2519 C
2520 C
2521 NNAC    = # of section in the nacelle
2522 NNACP   = # of points in nth section
2523 C
2524 C
2525 C     Read the nacelle geometry
2526 READ(30,810)
2527 READ(30,820)
2528 READ(30,830)NNAC,NNACP
2529 C
2530 M=0
2531 40 CONTINUE
2532 C
2533 READ(30,840)
2534 M = M + 1
2535 C
2536 DO 100 K=1,NNACP
2537 READ(30,850) XNAC(M),YNAC(M,K),ZNAC(M,K)
2538 YT(K)=YNAC(M,K)
2539 ZT(K)=ZNAC(M,K)
2540 100 CONTINUE
2541 CALL DISTARCYT,YT,NNACP,YT,ZT,NNACP,-10.,0
2542 DO 110 K=1,NNACP
2543   YM(K)=YT(K)
2544   ZM(K)=ZT(K)
2545 110 CONTINUE
2546 IF(M.LT.NNAC) GOTO 40
2547 C
2548 C     Write the nacelle geometry into PLOT3D Planar format
2549 KK=1
2550 KK=NNACP
2551 WRITE(52)KK,KW,M
2552 DO 800 L=1,M
2553   WRITE(52)(XNAC(L),K=1,KK),
2554   (YM(L,K),K=1,KK),
2555   (ZNAC(L,K),K=1,KK)
2556 800 CONTINUE
2557 C
2558 call flush (52)
2559
2560 810 FORMAT(1X)
2561 820 FORMAT(1X)
2562 830 FORMAT(3IX,I5,8X,I5/)
2563 840 FORMAT(1X)
2564 850 FORMAT(2X,3F16.7)
2565
2566 RETURN
2567 END
```

UNIX™  
FORTRAN Program

SOURCE PROGRAM  
**samgrid.f**

DATE	7/07/94
TIME	4:18:56 pm

PAGE #  
**39**

LINE #

SOURCE TEXT

```

2568 SUBROUTINE WINGMAKER
2569 C include "sgrid.com"
2570 C
2571 C This program generates a 'clipped' delta wing with no twist
2572 C based on airfoil coordinates read in from fort.90. To use as
2573 C part of samgrid, WINGIN is not necessary (nor is VARISWEEP).
2574 C
2575 C Written by: Donovan L. Mathias
2576 C : July 1992
2577 C
2578 C Whenever possible, the same variables are used as in samgrid.f
2579 C fort.90 airfoil coordinates
2580 C fort.77 description of the wing
2581 C
2582 C
2583 C Declarations
2584 C
2585 REAL XTE(LS),SLOPE1,SLOPE2,SCALE,XAF(150)
2586 REAL YU(150),YL(150)
2587 real SPAN
2588 INTEGER L,I,J,K
2589 C
2590 C Initialization
2591 C
2592 C read(77,*),NC
2593 C read(77,*),XLE(1),XLE(NC)
2594 C read(77,*),XTE(1),XTE(NC)
2595 C read(77,*),ZBASE(1,1,1),ZBASE(1,1,NC)
2596 C
2597 C Read in airfoil coordinates (L is # of X coords.)
2598 C
2599 C
2600 READ(90,*)
2601 READ(90,*)
2602 READ(90,*),NU
2603 READ(90,*)
2604 NL=NU
2605 DO I=1,NU
2606 READ(90,19)XAF(I),YU(I),YL(I)
2607 ENDDO
2608 19 format(3x,f9.7,3x,f9.7,3x,f9.7)
2609 C Establish Z distance (Spanwise)
2610 C
2611 C SPAN = ZBASE(1,1,NC)- ZBASE(1,1,1)
2612 C
2613 C DO K=0,NC-1
2614 C DO I=1,NU
2615 C ZBASE(I,1,K+1) = (K*(SPAN/(NC-1)))
2616 C ZBASE(I,2,K+1) = (K*(SPAN/(NC-1)))
2617 C ENDDO
2618 C
2619 C ENDDO
2620 C Establish Sweep (1 FOR LE, 2 FOR TE)
2621 C
2622 C SLOPE1 = (XLE(NC)-XLE(1))/(ZBASE(1,1,NC)-ZBASE(1,1,1))
2623 C SLOPE2 = (XTE(NC)-XTE(1))/(ZBASE(1,1,NC)-ZBASE(1,1,1))
2624 C
2625 C Generate leading and trailing edges
2626 C
2627 C DO K=1,NC
2628 C XLE(K) = XLE(1) + SLOPE1*ZBASE(1,1,K)
2629 C XTE(K) = XTE(1)+SLOPE2*(ZBASE(1,1,K)-ZBASE(1,1,1))
2630 C ENDDO
2631 C
2632 C Distribute grid points
2633 C
2634 C DO K=1,NC
2635 C SCALE = XTE(K)-XLE(K)
2636 C XBASE(1,1,K) = XLE(K) + SCALE*XAF(I)
2637 C DO I=1,NU
2638 C XBASE(1,1,K) = XLE(K) + SCALE*XAF(I)
2639 C YBASE(1,1,K) = YU(I)*SCALE
2640 C XBASE(1,2,K) = XLE(K) + SCALE*XAF(I)
2641 C YBASE(1,2,K) = YL(I)*SCALE
2642 C ENDDO
2643 C
2644 C Return values to original names
2645 C
2646 C DO K=1,NC
2647 C CHORD(K) = ABS(XLE(K)-XTE(K))
2648 C YLE(K) = YBASE(1,1,K)
2649 C ENDDO
2650 RETURN
2651 END
2652

```





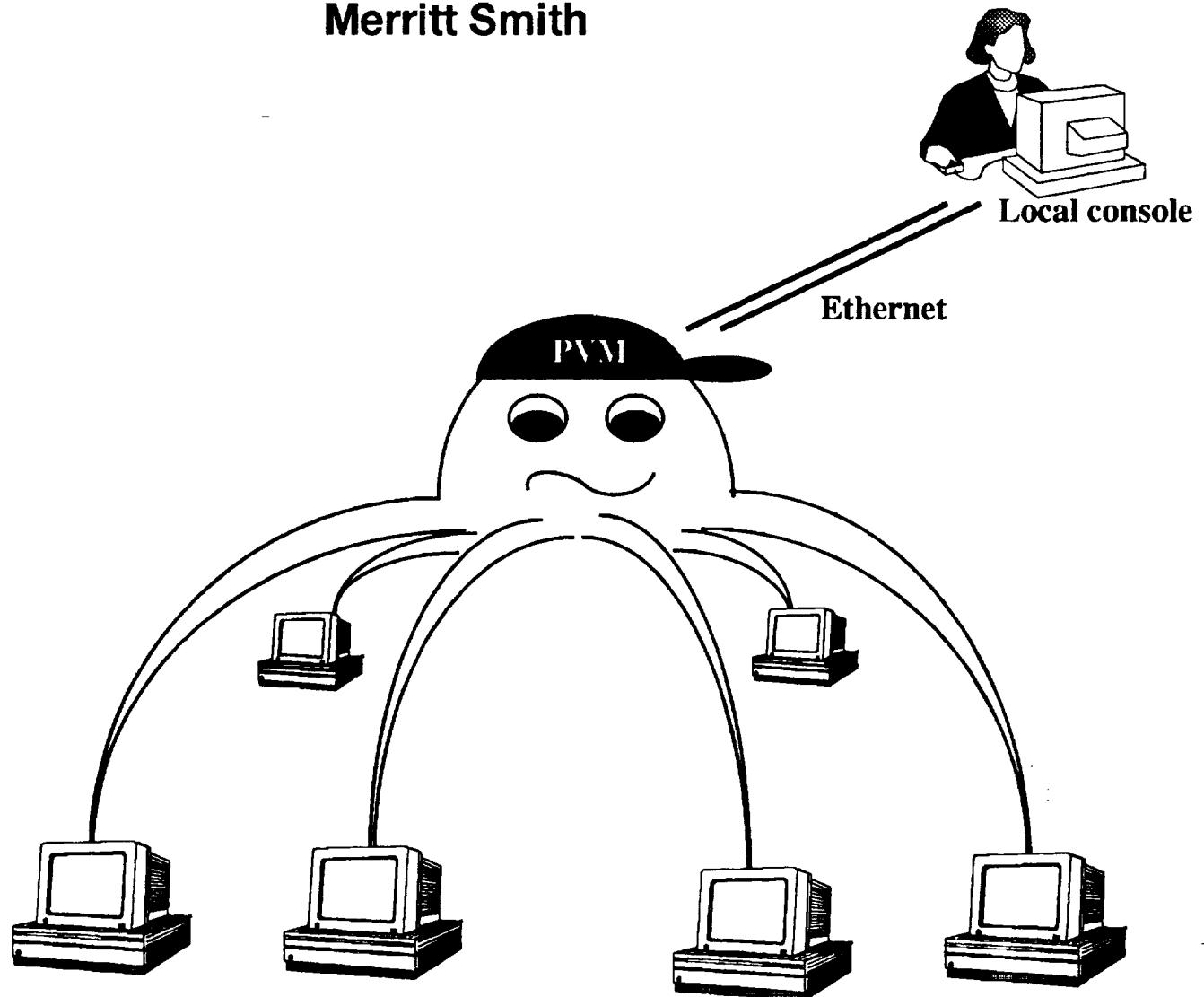
# **Appendix C**

## **PVM Manual**

# *Manual of PVM*

**Samson Cheung**

**Merritt Smith**





---

# *Table of Contents*

<b>1</b>	<b>Preface</b>	1
<b>2</b>	<b>Introduction</b>	
	Software Package.....	2
	Definitions .....	2
	Structure of PVM.....	3
	Directory Setup .....	3
<b>3</b>	<b>Programming Concepts</b>	
	Master and Slaves .....	4
	Single Program-Multiple Data (SPMD) .....	6
<b>4</b>	<b>PVM Daemon</b>	
	Console Commands .....	10
	Console Usage .....	11
<b>5</b>	<b>PVM Library</b>	
	Process Control .....	12
	Dynamic Configuration.....	13
	Message Buffers .....	13
	Packing and Unpacking .....	13
	Sending and Receiving .....	14
<b>6</b>	<b>Tutorial</b>	
	Golden Section .....	16
	Serial Program.....	17
	PVM Master Guideline/Master Program .....	18
	PVM Slave Guideline/Slave Program .....	22
	Compilation and Running.....	24

---

	Makefile .....	25
--	----------------	----

## **7 Problems and Tips**

	Problems .....	26
	Host File .....	27

## **8 Appendix**

	ARCH Names.....	29
	Error Codes.....	30

---

## *Preface*

This manual serves as a supplementary document for the official reference manual of a relatively new research software, PVM, which has been developed at Oak Ridge National Laboratory. A beginner, who has no previous experience with PVM, would find this manual useful.

We would like to thank you in advance that if you find any problems in PVM or this manual, please contact one of us.

Mr. Merritt Smith  
NASA Ames Research Center, MS 258-1  
Moffett Field, CA 94035  
e-mail: [mhsmit@nas.nasa.gov](mailto:mhsmit@nas.nasa.gov)  
phone: (415) 604-4462

Dr. Samson Cheung  
MCAT Institute  
NASA Ames Research Center, MS 258-1  
Moffett Field, CA 94035  
e-mail: [cheung@nas.nasa.gov](mailto:cheung@nas.nasa.gov)  
phone: (415) 604-4462

# 1 INTRODUCTION ■

---

This manual provides you with an introduction to PVM and provides the fundamentals necessary to write FORTRAN programs in the PVM environment through a tutorial sample. This manual is designed for those who have no previous experience with PVM. However, you should know basic FORTRAN programming and UNIX. If you are ready for an advanced PVM application, please consult the official PVM Reference Manual.

## Software Package

PVM stands for Parallel Virtual Machine. It is a software package that allows a heterogeneous network of parallel and serial computers to appear as a single concurrent computational resource. PVM allows you to link up all or some of the computational systems on which you have accounts, to work as a single distributed-memory (parallel) machine. We call this a Virtual Machine.

PVM is useful for the following reasons. Unlike large mainframe computers or vector supercomputers, workstations spend most of the time idle. The idle time on a workstation represents a significant computational resource. PVM links these workstations up to become a powerful multi-processor computational machine. With PVM, the lack of supercomputer resources should not be an obstacle to number crunching computational programs. Furthermore, the annual maintenance costs of a vector supercomputer is often sufficient to purchase the equivalent computing resource in the form of workstation CPU's.



## Definitions

Here are some terms we use throughout this document:

<i>Virtual Machine</i>	PVM links different user-defined computers together to perform as one large distributed-memory computer. We call this computer the Virtual Machine.
<i>Host</i>	Individual computer (member) in the virtual machine.
<i>Process</i>	Individual program operating on different computers or hosts.
<i>Processor</i>	The processing unit in computers. A virtual machine can be viewed as a multi-processor computer.

<i>Task</i>	The unit of computation handled by the virtual machine. You may want to think of one processor handling one task.
<i>Tid</i>	Task identification number which is a unique number used by the daemon and other tasks.
<i>Console</i>	A program from which you can directly interact with the virtual machine. (Add hosts, kill processes,...)

## Structure of PVM

The PVM software is composed of two parts. The first part is a daemon. We call it *pvm3d*. This is the control center of the virtual machine. It is responsible for starting processes, establishing links between processes, passing messages, and many other activities in PVM. Since the daemon runs in the background, you have to use PVM console to directly interact with the virtual machine.

The second part of the system is a library of PVM interface routines located in *libpvm3.a*. This library contains user callable routines for message passing, spawning processes, coordinating tasks, and modifying your machine. In writing your application, you will need to call the routines in this library.

## Directory Setup

This setup is for NAS system. Before you use PVM, you need to set up the following directories on *all* the machines that you want PVM to link:

- Make a directory *\$HOME/pvm3/bin/ARCH* in all the hosts of the virtual machine.



Note *ARCH* is used throughout this manual to represent the architecture name that PVM uses for a given computer. The table in the Appendix lists all the *ARCH* names that PVM supports. For example, for Silicon Graphic IRIS workstations, you should make a directory *\$HOME/pvm3/bin/SGI*.

- Make a directory *\$HOME/pvm3/include*, and copy the file *f\_pvm3.h* from */usr/nas/pkg/pvm3.2/include*. (If you are on different system from NAS, please consult your system consultant.)
- Make a directory *\$HOME/pvm3/codes*, and write your application programs in this directory. You can actually put your programs anywhere you like as long as the correct "include" files are includes. The current setup is for clarity.

## 2 *Programming Concept*

---

Unlike graphical software or a word-processor, you cannot *see* PVM working by clicking your mouse buttons. In fact, a virtual machine is quite an abstract concept because you don't physically have a multi-processor machine! In this chapter, you will learn a simple concept, which will help you to visualize how PVM works.

### Master and Slaves

A common way to work with PVM is a *Master/Slave* relationship. A Master process starts Slave routines and distributes work. However, a Master does not actively participate in the computation. A Master process most often resides on the originating host (user's computer), while the Slave programs are distributed to the hosts of the virtual machine.

You need to distribute executables of Slave programs to the directory `$HOME/pvm3/bin/ARCH` on every host. You can locate this Master program anywhere you like.

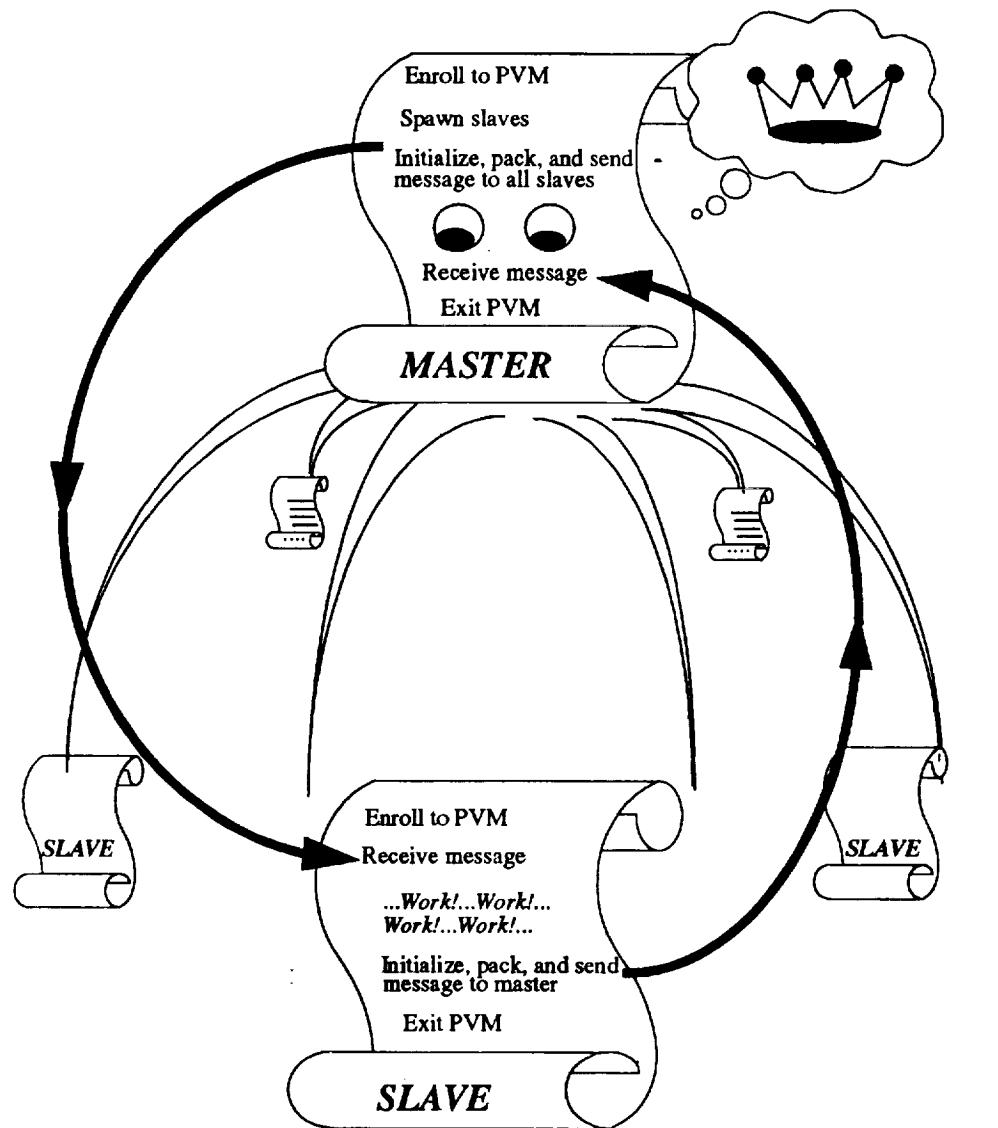
Since the Master program spawns Slave programs on each of the hosts to do jobs, it is important to understand the communication (message passing) among the hosts in PVM.

Typically, a Master and a Slave have the following logic:

	Master	Slave
1	Enroll itself to PVM	1 Enroll itself to PVM
2	Spawn slave processes	2 Receive message from master
3	Initialize buffer, pack, and send message to all slaves.	3 ...do something useful...
4	...wait for slaves to finish...	4 Initialize buffer, pack, and send message to master
5	Receive message from slave(s)	5 Exit PVM
6	Exit PVM	

The figure on the opposite page graphically describes a Master/Slave relationship and shows the exchange of information.

**FIGURE 1.** Communication in *Master/Slave* programs.



## SPMD

Another common way to work with PVM is the **SPMD**, Single Program Multiple Data model. There is only a single program, and there is no Master program directing the computation. The user starts the first copy of the program and using the routine `pvmfparent()`, this copy can determine that it was not spawned by PVM, and thus must be the first copy (parent). It then spawns multiple copies (children) of itself and passes them the array of *tids*. At this point each copy is equal and can work on its partition of the data in collaboration with the other processes.

Typically, a SPMD program has the following logic:

1. **Enroll in pvm**
2. **If I am the first copy (parent)**
  - a) Spawn child processes
  - b) Initialize buffer, pack, and send message out
3. **If I am a secondary copy (child)**  
Receive messages
4. **Work!...Work!...Work!**
5. **Exit PVM**

The program on the opposite page describes a SPMD logic and shows the exchange of information. Please spend some time to study the program.

In the next chapter we will introduce the PVM daemon and the fundamentals of message passing.

## SPMD Program

```

c-----+
c   SPMD Fortran example using PVM 3.0
c-----+
c
      program spmd
      include '../include/fpvmp3.h'
      PARAMETER( NPROC=4 )
      integer mytid, me, i
      integer tids(0:NPROC)

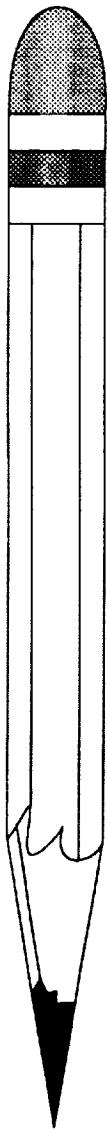
      c Enroll in pvm
      call pvmfmytid( mytid )

1      c -----
      c Find out if I am parent or child
      c -----
      call pvmfparent(tids(0))
      if( tids(0) .lt. 0 ) then
          tids(0) = mytid
          me = 0
      -----
      c start up copies of myself
      -----
      call pvmfspawn('spmd',PVMDEFAULT,'*',
* NPROC-1,tids(1), info)
      -----
      c multicast tids array to children
      -----
      call pvmfinitsend( PVMDEFAULT, info )
      call pvmfpack( INTEGER4, tids, NPROC, 1, info )
      call pvmfmcast( NPROC-1, tids(1), 0, info )
      else
      -----
      c receive the tids array and set me
      -----
      call pvmfrecv( tids(0), 0, info )
      call pvmfunpack( INTEGER4, tids, NPROC,1,info)
      do 30 i=1, NPROC-1
          if( mytid .eq. tids(i) ) me = i
30      continue
      endif
      c -----
      c all NPROC tasks are equal now
      c and can address each other by tids(0) thru tids(NPROC-1)
      c for each process me => process number [0-(NPROC-1)]
      c -----
      print*,'me =',me, ' mytid =',mytid
      call dowork( me, tids, NPROC )

4      c -----
      c program finished exit pvm
      c -----
      call pvmfexit(info)
      stop
      end

```

*Notes*



```
subroutine dowork( me, tids, nproc )
  include '../include/fpvm3.h'
c-----
c Simple subroutine to pass a token around a ring
c-----
  integer me, nproc
  integer tids( 0:nproc)

  integer token, dest, count, stride, msgtag

  count = 1
  stride = 1
  msgtag = 4

  if( me .eq. 0 ) then
    token = tids(0)
    call pvmfinitsend( PVMDEFAULT, info )
    call pvmfpack( INTEGER4,token,count,stride,info)
    call pvmfsend( tids(me+1), msgtag, info )
    call pvmfrecv( tids(nproc-1), msgtag, info )
    print*, 'token ring done'
  else
    call pvmfrecv( tids(me-1), msgtag, info )
    call pvmfunpack( INTEGER4,token,count,stride,info)
    call pvmfinitsend( PVMDEFAULT, info )
    call pvmfpack( INTEGER4,token,count,stride,info)
    dest = tids(me+1)
    if( me .eq. nproc-1 ) dest = tids(0)
    call pvmfsend( dest, msgtag, info )
  endif

  return
end
```

# 3

## *PVM Daemon*



---

The PVM daemon is the control center of the virtual machine. You can activate the PVM daemon by starting the PVM console or by invoking the daemon directly with a list of hosts. The latter will be discussed in chapter 6. To start the console, enter `pvm` at UNIX prompt on your local machine. The PVM console prints the prompt

`pvm>`

and accepts commands from standard input. The console allows interactive adding and deleting of hosts to the virtual machine as well as interactive starting and killing of PVM processes. Even if the daemon is started directly, the console can be used to modify the virtual machine.

## Console Commands

Here are the commands available in the PVM console:

ADD	add other computers (hosts) to PVM
ALIAS	define and list command aliases you set
CONF	show members in virtual machine
DELETE	remove hosts from pvm
ECHO	echo arguments
HALT	stop all pvm processes and exit deamon
HELP	print this information
ID	print console task identity
JOBS	display list of running jobs
KILL	terminate tasks
MSTAT	show status of hosts
PS	list tasks
PSTAT	show status of tasks
QUIT	exit PVM console, but PVM daemon is still activated
RESET	kill all tasks
SETENV	display or set UNIX environment variables
SIG	send signal to task
SPAWN	spawn task
UNALIAS	remove alias commands you previous set
VERSION	show PVM version

## Console Usage

Suppose the console is running on workstation *win210*. This computer will automatically be a host in your virtual machine. Here are some examples of using PVM console:

**1. Activate PVM console**

```
win210> pvm
```

**2. Add amelia and fred to your virtual machine**

```
pvm> add amelia
```

```
1 successful
```

HOST	DTID
amelia	c0000

```
pvm> add fred
```

```
1 successful
```

HOST	DTID
fred	100000

**3. Check the configuration of your virtual machine**

```
pvm> conf
```

```
3 host, 1 data format
```

HOST	DTID	ARCH	SPEED
win210	40000	SGI	1000
amelia	c0000	SGI	1000
fred	100000	SGI	1000

**4. Delete amelia**

```
pvm> delete amelia
```

```
1 successful
```

HOST	STATUS
amelia	deleted

**5. Exit PVM console, but PVM daemon is still running**

```
pvm> quit
```

```
pvmd still running
```

```
win210>
```

**4***PVM Library*

## Process Control

**call pvmfmytid( tid )**

This routine enrolls this process with the PVM daemon on its first call, and generates a unique **tid**. You call this routine at the beginning of your program.

**call pvmfexit( info )**

This routine tells the local PVM daemon that this process is leaving PVM. You call this routine at the end of your program. Values of **info** less than zero indicate an error.

**call pvmfkill( tid, info )**

This routine kills a PVM task identified by **tid**. Values of **info** less than zero indicate an error.

**call pvmfspawn( pname,flag,where,ntask,tids,numt )**

This routine starts up **ntask** instances of a single process named **pname** on the virtual machine. Here are the definition of the other arguments:

<b>flag</b>	<b>Option Value</b>	<b>Meaning</b>
	PVMDEFAULT (0)	PVM can choose any machine to start task
	PVMHOST (1)	where specifies a particular host
	PVMARCH (2)	where specifies a type of architecture
	PVMDEBUG (4)	start up processes under debugger
	PVMTRACE (8)	processes will generate PVM trace data
<b>where</b>	is where you want to start the PVM process. If <b>flag</b> is 0, <b>where</b> is ignored.	
<b>tids</b>	contains identification numbers of PVM processes started by this routine.	
<b>numt</b>	indicates how many processors started; negative values indicate an error.	

**Note** You should always check **tids** and **numt** to make sure all processes started correctly.



**call pvmfparent( tid )**

This routine returns the **tid** of the process that spawned this task. If the calling process was not created with **pvmfspawn**, then **tid=PvmNoParent**.

## Dynamic Configuration

**call pvmfaddhost( host, info )**

**call pvmfdelhost( host, info )**

These routines add and delete hosts to the virtual machine respectively. Values of **info** less than zero indicate an error.



**Note** Both routines are expensive operations that require the synchronization of the virtual machine.

## Message Buffers

**call pvmfinitsend( encoding, bufid )**

This routine clears the send buffer, and creates a new one for packing a new message.

**encoding**

**Encoding Value**

**Meaning**

PVMDEFAULT (0)

XDR encoding if virtual machine configuration is heterogeneous

PVMRAW (1)

no encoding is done. Messages are sent in their original format.

PVMINPLACE (2)

data left in place. Buffer only contains sizes and pointers to the sent items.

**bufid** contains the message buffer identifier. Values less than zero indicate an error.



This is not implemented in PVM v3.2.

**call pvmffreebuf( bufid, info )**

This routine disposes the buffer with identifier **bufid**. You use it after a message has been sent, and is no longer needed. Values of **info** less than zero indicate an error.

## Packing and Unpacking

**call pvmfpack( what, xp, nitem, stride, info )**

**call pvmfunpack( what, xp, nitem, stride, info )**

These routines pack/unpack your message **xp**, which can be a number or a string. You can call these routines multiple times to pack/unpack a single message. Thus a message can contain several arrays, each with a different data type.



- Note** There is no limit to the complexity of the packed messages, but you must unpack them exactly as they were packed.
- what** indicates what type of data **xp** is
- |              |               |
|--------------|---------------|
| STRING (0)   | REAL (4)      |
| BYTE1 (1)    | COMPLEX8 (5)  |
| INTEGER2 (2) | REAL8 (6)     |
| INTEGER4 (3) | COMPLEX16 (7) |
- nitem** is number of items in the pack/unpack. If **xp** is a vector of 5, **nitem** is 5.
- stride** is the stride to use when packing.
- info** is status code returned by this routine. Values less than zero indicate an error.

## Sending and Receiving

**call pvmfsend( tid, msgtag, info )**

This routine labels the message with an integer identifier **msgtag**, and sends it immediately to the process **tid**. Values of **info** less than zero indicate an error.

**call pvmfmcast( ntask, tids, msgtag, info )**

This routine labels the message with an integer identifier **msgtag**, and broadcasts the message to all **ntask** number of tasks specified in the integer array **tids**. Values of **info** less than zero indicate an error.

**call pvmfreccv( tid, msgtag, bufid )**

This routine blocks the flow of your program until a message with label **msgtag** has arrived from **tid**. A value of -1 in **msgtag** or **tid** matches anything (wildcard). This routine creates a new active receive buffer, and puts the message in it. Values of **bufid** identify the newly created buffer; values less than zero indicate an error.

**call pvmfnrecv( tid, msgtag, bufid )**

This routine performs in the same way as **pvmfreccv**, except that it does not block the flow of your program. If the requested message has not arrived, this routine returns **bufid=0**. This routine can be called multiple times for the same message to check if it has arrived, while performing useful work between calls. When no more useful work can be performed, the blocking receive **pvmfreccv** can be used for the same message.

**call pvmfprobe( tid, msgtag, bufid )**

This routine checks if a message has arrived; however, it does not receive the message. If the requested message has not arrived, this routine returns **bufid=0**. This routine can

be called multiple times for the same message to check if it has arrived, while performing useful work between calls.

**call pvmfbufinfo (bufid, bytes, msgtag, tid, info)**

This routine returns information about the message in the buffer identified by **bufid**. The information returned is the actual **msgtag**, source **tid**, and message length in **bytes**. Values of **info** less than zero indicate an error.

# 5 Tutorial

This chapter shows you how PVM may be applied to your application programs through a simple example. The example chosen is the Golden Section rule for finding the maximum of a function. You may remember it from Math class in high school. Let us review the method and the algorithm.

## Golden Section

Suppose we want to find the maximum of a curve  $y=f(x)$ ; where  $x$  is between the interval  $a_1$  and  $a_2$ . The points  $a_3$  and  $a_4$  are symmetrically placed in this interval, so that

$$a_3 = (1-\alpha) a_1 + \alpha a_2 \quad (\text{EQ 1})$$

$$a_4 = \alpha a_1 + (1-\alpha) a_2 \quad (\text{EQ 2})$$

See Figure 1 at left. Golden Section rule requires  $\alpha$  to be 0.382.

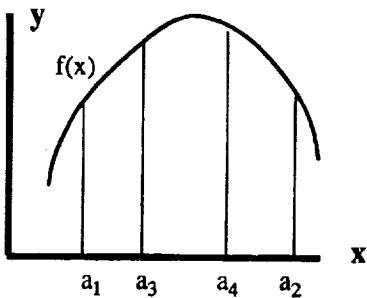


Figure 1. Interval division for Golden Section

The algorithm of finding the maximum is as follow:

If $f(a_4) < f(a_3)$	If $f(a_4) > f(a_3)$
1 Consider new interval $(a_1, a_4)$	1 Consider new interval $(a_3, a_2)$
2 Apply EQ.(1) and (2) again	2 Apply EQ. (1) and (2) again
3 Until maximum is reached	3 Until maximum is reached
If $f(a_3)=f(a_4)$ , the maximum is found	

The FORTRAN program (Serial Program) on the opposite page is the Golden Section rule that a programmer would write on a normal serial computer. Please spend a few minutes to study the flow of the program. This simple program consists of two parts, the main (calling) program and the function subroutine. The latter has only four lines.



Note Notice that for each interval  $(a_1, a_2)$ , we need to call the function evaluation four times to find  $f(a_1)$ ,  $f(a_2)$ ,  $f(a_3)$ , and  $f(a_4)$ .

# Serial Program

```

C      Linear optimization:
C
C      Search for maximum of a x-y curve.
C
DIMENSION A(4),FN(4)
C
C      Initial interval
L = 0
TOL = 1.E-3
A(1) = 0.4
A(2) = 1.6
Golden Section rule → ALPHA = 0.382
C
C
10    CONTINUE
C
C      Loop begins:
L = L + 1

Equations (1) and (2) →
A(3) = (1.-ALPHA)*A(1) + ALPHA*A(2)
A(4) = ALPHA*A(1) + (1.-ALPHA)*A(2)
FN(1) = F(A(1))
FN(2) = F(A(2))
FN(3) = F(A(3))
FN(4) = F(A(4))
WRITE(10,*)'A  ',A(1),A(2),A(3),A(4)
WRITE(10,*)'F  ',FN(1),FN(2),FN(3),FN(4)
WRITE(10,*)' '
ERR = ABS(FN(2)-FN(3))
IF(ERR.LE.TOL) GOTO 999
C
C
IF(FN(4) .GT. FN(3)) THEN
  B1 = A(3)
  B2 = A(2)
  A(1) = B1
  A(2) = B2
  GOTO 10
ELSEIF(FN(4) .LT. FN(3)) THEN
  B1 = A(1)
  B2 = A(4)
  A(1) = B1
  A(2) = B2
  GOTO 10
ENDIF
999  CONTINUE
STOP
END

Function evaluation →
FUNCTION F(X)
F = TANH(X)/(1.+X*X)
RETURN
END

```

## PVM Master Guideline

Recall that in the procedure of finding a new interval, the program calls the function evaluation four times *serially* to get  $f(a_1)$ ,  $f(a_2)$ ,  $f(a_3)$ , and  $f(a_4)$ . We would like to assign four processors to perform the four function evaluations *simultaneously* on the virtual machine. Therefore, we modify the Serial Program by writing the main (calling) program as a Master program, and the function subroutine as a Slave program.

The following steps are general guidelines to writing a Master program. Please study the steps, and compare them with the program on the opposite page. Also compare it with the Serial Program.

### 1. Include fpvm3.h

Include this file in your program, you are able to use the PVM preset variables; such as PVMDEFAULT, REAL4, and more, mentioned in Chapter 4 and the Appendix.

### 2. Enroll Master to PVM

Use `pvmfmytid(mytid)` to enroll.

### 3. Assign virtual processors

Use the following call to spawn `nproc` function processes.

`pvmfspawn(pname, PVMDEFAULT, where, nproc, tids, numt)`

Also tell PVM the name of the Slave program (`pname`). PVM returns `tids`, the identifier of the `nproc` processors.

### 4. Initialize buffer and pack data

Use `pvmfinitsend` to clear buffer.

Use the following routine to pack a real array `A` of dimension `m`.

`pvmfpack(REAL4, A, m, 1, info)`

### 5. Send message

Use the following call to send the packed message to the Slave process identified by `tids`.

`pvmficast(nproc, tids, msgtag, info)`

# Master Program

```

C      Linear optimization:
C      Search for maximum of a x-y curve.
PROGRAM MASTER

C
C
I   include '../include/fpvm3.h'
DIMENSION A(4),FN(4)
integer tids(0:32),who
character*8 where
character*12 pname

2   c      Enroll this program in PVM
c      call pvmfmytid(mytid)
c      Start up the four processors
nproc = 4
where = '*'
pname = 'function'
call pvmfspawn(pname,PVMDFAULT,where,nproc,tids,numt) Assign four processors
Slave program's name
do 20 i=0,nproc-1
      write(*,*) 'tid', i, tids(i)
20   continue

C
C      Initial interval
L = 0
A(1) = 0.4
A(2) = 1.6
ALPHA = 0.382
TOL = 1.E-3
ERR = 1.

C
10   CONTINUE

C
C
C      Loop begins:
L = L + 1

Equations (1) and (2) →
A(3) = (1.-ALPHA)*A(1) + ALPHA*A(2)
A(4) = ALPHA*A(1) + (1.-ALPHA)*A(2)

C
C      Broadcast data to all node programs
C      first pack them, then send them
call pvmfinitsend(PVMDFAULT,info)
call pvmfpack(INTEGER4,nproc,1,1,info)
call pvmfpack(INTEGER4,tids,nproc,1,info)
call pvmfpack(REAL4,A,4,1,info)
call pvmfpack(REAL4,ERR,1,1,info)

4   Pack nproc, tids, A,
     and ERR
C
C
5   msgtype = 1
     call pvmfcast(nproc,tids,msgtype,info)
msgtype value matches the one
received in Slave program
C

```

**6. Wait until messages come from Slaves**

Use `pvmfrecv()` to block until Slaves return function values.  
Make sure value of `msgtype` matches values coming from Slaves.

**7. Receive and Unpack data**

The sequence of unpacking is the same as the packing in the Slave.

**8. Exit PVM**

Use `pvmfexit(info)` to exit PVM.

6

c Wait for results from processors

msgtype value matches the  
one sent from Slave program

7

Receive /unpack FN and 'who'  
from the 4 processors one by one

```
msgtype = 2
do 100 i=1,nproc
    call pvmfrecv(-1,msgtype,info)
    call pvmunpack(INTEGER4,who,1,1,info)
    call pvmunpack(REAL4,FN(who),1,1,info)
    continue

    WRITE(10,*) 'A  ',A(1),A(2),A(3),A(4)
    WRITE(10,*) 'F  ',FN(1),FN(2),FN(3),FN(4)
    WRITE(10,*) ' '
    ERR = ABS(FN(2)-FN(3))
    IF(ERR.LE.TOL) GOTO 999
```

C

C

```
IF(FN(4) .GT. FN(3)) THEN
    B1 = A(3)
    B2 = A(2)
    A(1) = B1
    A(2) = B2
    GOTO 10
ELSEIF(FN(4) .LT. FN(3)) THEN
    B1 = A(1)
    B2 = A(4)
    A(1) = B1
    A(2) = B2
    GOTO 10
ENDIF
```

C

```
c Program finished leave PVM before exiting
999 continue
```

```
call pvmfexit(info)
STOP
END
```

8

---

## PVM Slave Guideline

The Slave program is basically the function evaluation program. In order to do the function evaluation, it needs information from Master. For example, it needs the identity numbers (`tids(1), ..., tids(4)`) that PVM assigns, and the values of  $a_1, \dots, a_4$ .

The following steps are general guidelines to writing a Slave program. Please study the steps, and compare them with the program on the opposite page. Also try to find the connection with the Master Program. You may find Figure 1 helpful.

**1. Include fpvm3.h**

Include this file in your program, you are able to use the PVM preset variable names; such as `PVMDEFAULT`, `REAL4`, and more, mentioned in all tables in Chapter 4 and the Appendix.

**2. Enroll Slave with PVM**

Use `pvmfmytid(mytid)` to enroll.

**3. Identify the parent of this process**

Use the following call to obtain the task identifier (`mtid`) of parent process. This is useful for returning solutions to the Master.  
`pvmfparent(mtid)`

**4. Receive and Unpack data**

Make sure the value of `msgtype` matches the one from Master. The sequence of unpacking is the same as the packing in Master.

**5. Perform function evaluation**

**6. Initialize buffer and pack data**

Use `pvmfinitsend` to clear buffer.

Use the following call to pack a real array `F` of dimension `n`.

`pvmfpack(REAL4, F, n, 1, info)`

**7. Send data**

Use the following call to send the packed message to Master:

`pvmfsend(mtid, msgtag, info)`

**8. Exit PVM**

Use `pvmfexit(info)` to exit PVM.

## Slave Program

```

c      program function
c
c
c      include '../include/fpvm3.h'

1      integer tids(0:32),who
      real a(32)
      tor = 1.e-3

2      c      Enroll this program in PVM
      call pvmfmytid(mytid)
      c      Get the parent's task id
      call pvmfparent(mtid)

3      c      continue

4      c      Receive data from host
      msgtype = 1
      call pvmfrecv(mtid,msgtype,info)
      call pvmfunpack(INTEGER4,nproc,1,1,info)
      call pvmfunpack(INTEGER4,tids,nproc,1,info)
      call pvmfunpack(REAL4,A,4,1,info)
      call pvmfunpack(REAL4,ERR,1,1,info)

      c
      c      if(err.le.tor) go to 99

5      c      Determine which processor I am
      do 5 i=0,nproc-1
          if(tids(i).eq.mytid) me = i
      5    continue
      who = me + 1

5      Function
      evaluation      c      Calculate the function
      c      X = A(who)
      c      f = TANH(X)/(1.+X*X)

6      Pack f and
      processor 'who'
      c      Send the result to Master
      call pvmfinitsend(PVMDEFAULT,info)
      call pvmfpack(INTEGER4,who,1,1,info)
      call pvmfpack(REAL4,f,1,1,info)
      msgtype = 2
      call pvmfsend(mtid,msgtype,info)
      go to 3

7      Go to 3 and wait for
      another call from master
      c      Program finished. Leave PVM before exiting
      99    continue
      call pvmfexit(info)
      stop
      end

```

## Compilation and Running

After you finish your program, it is time to compile and run. Follow the steps below to compile your programs.

**1. Make sure you have the correct directory setup**

Follow the advice from *Directory Setup* in Chapter 1.

**2. Compile the program**

Use the sample `Makefile` on the opposite page to compile your programs.



**Note** The `Makefile` links the PVM library, `libfpvm3.a`.

**3. Copy executables to all the hosts**

Follow the advice from *Directory Setup* in Chapter 1, and distribute the executables to `$HOME/pvm3/bin/ARCH`.

**4. Activate PVM**

Activate PVM by entering `pvm` at UNIX prompt.

**5. Decide the configuration of the virtual machine**

Add or delete hosts to the virtual machine. (Chapter 3)

**6. Quit PVM console**

Leave PVM console (don't halt daemon) by entering `quit` at the `pvm` prompt.

## Makefile

```
#  
# Custom section  
# Set PVM_ARCH to your architecture type (SUN4, HP9K, RS6K, # SGI,  
etc.)  
# if PVM_ARCH = BSD386 then set ARCHLIB = -lrpc  
# if PVM_ARCH = SGI      then set ARCHLIB = -lsun  
# if PVM_ARCH = I860     then set ARCHLIB = -lrpc -lsocket  
# if PVM_ARCH = IPSC2    then set ARCHLIB = -lrpc -lsocket  
# otherwise leave ARCHLIB blank  
#  
# PVM_ARCH and ARCHLIB are set for you if you use 'aimk'.  
#  
PVM Library _____  
PVM_LIB          =      SGI  
ARCHLIB          =      -lsun  
# END of custom section - leave this line here  
#  
PVMDIR   =      /amd/fs02/pub/iris4d_irix4/nas/pkg/pvm3.2  
PVMLIB   =      $(PVMDIR)/lib/$(PVM_ARCH)/libpvm3.a  
SDIR      =      .  
BDIR      =      /u/wk/cheung/pvm3/bin  
XDIR      =      $(BDIR)/$(PVM_ARCH)  
  
CFLAGS   =      -g -I../include  
LIBS     =      $(PVMLIB) $(ARCHLIB)  
  
F77      =      f77  
FFLAGS   =      -g  
FLIBS    =      $(PVMDIR)/lib/$(PVM_ARCH)/libfpvm3.a $(LIBS)  
  
default:        master function  
  
$(XDIR):  
    - mkdir $(BDIR)  
    - mkdir $(XDIR)  
  
clean:  
    rm -f *.o bfgs quadfunct  
  
master: $(SDIR)/master.f  $(XDIR)  
        $(F77) $(FFLAGS) -o master $(SDIR)/master.f $(FLIBS)  
        mv master $(XDIR)  
  
function: $(SDIR)/function.f  $(XDIR)  
        $(F77) $(FFLAGS) -o function $(SDIR)/function.f $(FLIBS)  
        mv function $(XDIR)
```

# 6

# Problems and Tips ■

---

PVM is a relatively new piece of software. It is not advanced enough to warn you ahead of time before problems come. Here are a couple of cases that you may encounter as a beginner.

## Problems

### Can't activate PVM



- If the message you get, after entering `pvm` at UNIX prompt, is `libpvm [pid-1]: Console: Can't start pvm`, it is possible that the last time you halted PVM daemon, the daemon created a residual file `/tmp/pvmd.xxxx`; where `xxxx` is an unique number for you. Delete this file, and start PVM again.
- If the daemon is running but the PVM console will not start, it is possible that you have too many processes running. You have to kill all the processes before you re-activate PVM console.



Note Use `ps -ef | username` at UNIX prompt to locate your running processes.

### Can't add hosts



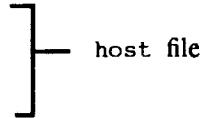
It is possible that there are no links between your local computer and the other hosts. Check the following two things:

- Make sure each of your hosts has a `.rhosts` file in the `$HOME` directory, and this file points to your local computer.
- Make sure the `.rhosts` file is "read" and "write" protected from others users.

## Host File

You can create the following file to build the virtual machine without activating the PVM console. The addresses must be recognizable by your system.

```
computer1.address  
computer2.address  
computer3.address  
computer4.address
```



Note The first machine listed must be the initiating host.

Note If tasks are to be spawned on specific systems, the system name contained in where (routine `pvm_spawn`) must match the name in the host file exactly.

Note If spawning tasks are on the initiating host, use the truncated host name. For example, if the full address is `win210.nas.nasa.gov` ; use `win210` instead. This is a bug in PVM v3.2.

Having the host file ready, enter the following at UNIX prompt,

```
win210> pvmmd3 host
```

---

**Problems and Tips**

Place to jot down problems.

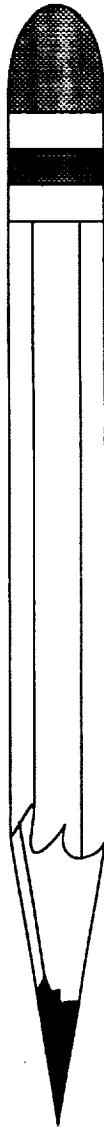
**Notes**

If encounter problems, please contact:

Merritt Smith: [mhsmith@nas.nasa.gov](mailto:mhsmith@nas.nasa.gov)

or

Samson Cheung: [cheung@nas.nasa.gov](mailto:cheung@nas.nasa.gov)



# *Appendix*

---

**TABLE 1. ARCH names used in PVM.**

---

<b>ARCH</b>	<b>Machine</b>	<b>Note</b>
AFX8	Alliant FX 8	
ALPHA	DEC Alpha	DEC OSF-1
BAL	Sequent Balance	DYNIX
BFLY	BBN Butterfly TC2000	
BSD386	80386/486 Unix box	BSDI
CM2	Thinking Machines CM2	Sun front-end
CMS	Thinking Machines CMS	
CNVX	Convex C-series	
CNVXN	Convex C-series	native mode
CRAY	C-90, YMP, Cray-2	UNICOS
CRAYSMP	Cray S-MP	
DGAV	Data General Aviion	
HP300	HP-9000 model 300	HPUX
HPPA	HP-9000 PA-RISC	
I860	Intel iPSC/860	link-lprc
IPSC2	Intel iPSC/860 host	SysV
KSR1	Kendall Square KSR-1	OSF-1
NEXT	NeXT	
PGON	Intel Paragon	link -lprc
PMAX	DECstation 3100,5100	Ultrix
RS6K	IBM/RS6000	AIX
RT	IBM RT	
SGI	Silicon Graphics IRIS	link -lsun
SUN3	Sun 3	SunOS
SUN4	Sun 4, SPARCstation	
SYMM	Sequent Symmetry	
TITN	Staedent Titan	
UVAX	DEC Micro VAX	

---

**TABLE 2. Error codes returned by PVM routines**

---

Error Code	Meaning
PvmOK (0)	All right
PvmBadParam (-2)	Bad parameter
PvmMismatch (-3)	Barrier count mismatch
PvmNoData (-5)	Read past end of buffer
PvmNoHost (-6)	No such host
PvmNoFile (-7)	No such executable
PvmNoMem (-10)	Can't get memory
PvmBadMsg (-12)	Can't decode received message
PvmSysErr (-14)	Pvmd not responding
PvmNoBuf (-15)	No current buffer
PvmNoSuchBuf (-16)	Bad message identifier
PvmNukkGroup (-17)	Null group name is illegal
PvmDupGroup (-18)	Already in group
PvmNoGroup (-19)	No group with that name
PvmNotInGroup (-20)	Not in group
PvmNoInst (-21)	No such instance in group
PvmHostFail (-22)	Host failed
PvmNoParent (-23)	No parent task
PvmNoImpl (-24)	Function not implemented
PvmDSysErr (-25)	Pvmd system error
PvmBadVersion (-26)	Pvmd-pvmd protocol mismatch
PvmOutOfRes (-27)	Out of resources
PvmDupHost (-28)	Host already configurated
PvmCantStart (-29)	Fail to execute new slave pvmd
PvmAlready (-30)	Slave pvmd already running
PvmNoTask (-31)	Task does not exist
PvmNoEntry (-32)	No such (group,instance)
PvmDupEntry (-33)	(Group,instance) already exists

---



